

Link Presentación:

[https://www.canva.com/design/DAGktqDHwOU/JhQHngOXalZv-IY4eYWLwQ/edit?utm\\_content=DAGktqDHwOU&utm\\_campaign=designshare&utm\\_medium=link2&utm\\_source=sharebutton](https://www.canva.com/design/DAGktqDHwOU/JhQHngOXalZv-IY4eYWLwQ/edit?utm_content=DAGktqDHwOU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

#### Alumno

<b>Alumno</b>	Martín Losada Canedo
---------------	----------------------

#### Datos do Proxecto

<b>Título</b>	Hardening avanzado en sistemas GNU/Linux e Windows: SELinux, AppLocker
---------------	--

#### Índice:

1.	<b>Obxectivo</b>
2.	<b>Descrición</b>
3.	<b>Medios a utilizar</b>
4.	<b>Execución GNU/Linux</b>
5.	<b>Execución Windows</b>
6.	<b>Bibliografía/Webgrafía</b>

## 1. Obxectivo

O obxectivo principal deste proxecto é diseñar, implementar e probar un sistema de hardening avanzado tanto en entornos Linux como Windows, co fin de reforzar a seguridade fronte a posibles ameazas. Aseguraremos a protección dos servizos críticos, como SSH, mediante a configuración de políticas restritivas e o uso de SELinux, e estableceremos controles de execución de aplicacións en Windows mediante AppLocker.

O proxecto busca demostrar a capacidade para:

- Aplicar técnicas de fortificación en sistemas operativos reais.
- Utilizar ferramentas específicas como SELinux e tamén auditd, Fail2Ban, nftables. En Windows, AppLocker.
- Crear políticas personalizadas de seguridade.
- Validar as configuracións mediante probas prácticas controladas.

Todo isto orientado a acadar un sistema robusto, cun acceso seguro, minimizando a superficie de ataque e garantindo a trazabilidade das accións dos usuarios.

## 2. Descripción

O proxecto estrutúrase en dúas partes principais:

- Hardening avanzado en Linux (Debian):

Realizouse unha fortificación profunda do servizo SSH, modificando a súa configuración por defecto e aplicando medidas de control de acceso, detección de intrusionés e reforzo dos permisos de ficheiros. Ademais, implementáronse políticas de SELinux para restrinxir e auditar o comportamento dos usuarios e procesos críticos. A protección completouse cun firewall baseado en nftables que controla o tráfico entrante e saínte, así como medidas adicionais como o uso de Port Knocking e 2FA.

- Control de execución en Windows Server mediante AppLocker:

Configurouse un dominio Active Directory para a xestión centralizada de políticas de seguridade. Creáronse regras de AppLocker para restrinxir a execución de programas, scripts e instaladores, asegurando que só aplicacións autorizadas poidan executarse nos equipos cliente. O control aplicouse mediante GPOs específicas e probouse nunha máquina Windows 10 Cliente unida ao dominio.

Cada cambio realizouse documentando os procedementos, configurando probas específicas para validar a súa eficacia, e asegurando que todos os mecanismos de seguridade fosen activos e operativos.

## 3. Medios a utilizar

### 1. MÁQUINAS VIRTUALES

- **Servidor Debian (Linux):**
  - Sistema operativo Debian 12.
  - Función principal: Servidor SSH fortificado mediante configuración manual e políticas de SELinux.
- **Cliente Kali Linux:**

- Sistema operativo Kali Linux actualizado.
- Función principal: Máquina de ataque para realizar probas de acceso, escaneo de portos e validación das medidas de hardening aplicadas no servidor Debian.
- **Servidor Windows Server 2019:**
  - Sistema operativo Windows Server 2019.
  - Funcións principais:
    - Controlador de dominio Active Directory.
    - Configuración e xestión de políticas de AppLocker para controlar a execución de aplicacións en equipos cliente.
- **Clientes Windows 10 Education:**
  - Sistema operativo Windows 10 Education.
  - Función principal: Equipos unidos ao dominio para aplicar e probar as políticas de restricción de AppLocker.

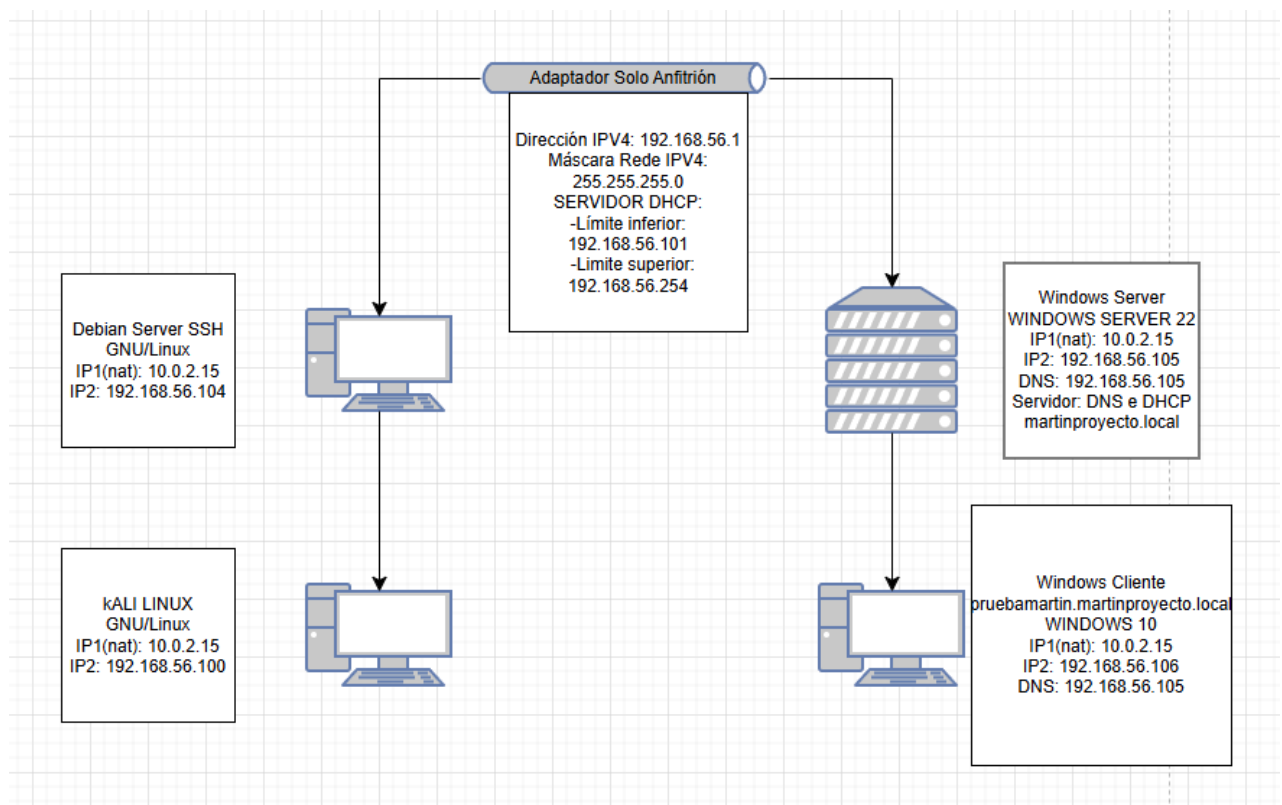
Cada máquina virtual foi configurada cunha rede adaptada ás necesidades específicas do proxecto: comunicación interna (Só Anfitrión) para o tráfico entre as máquinas, e NAT para acceso puntual a Internet ( se é necesario)

Pero podemos ter en conta as características de cada máquina virtual utilizada. Creei unha tabla para que poidades ver todo reflexado:

MÁQUINAS VIRTUALES				
	Windows Server	Debian Server	Cliente Win	Cliente Kali
Sistema Operativo	Windows Server 19	Debian 12	Windows 10	Kali Linux
Procesador y RAM	<b>CPU:</b> 1 <b>RAM:</b> 4gb	<b>CPU:</b> 2 <b>RAM:</b> 4gb	<b>CPU:</b> 1 <b>RAM:</b> 2gb	<b>CPU:</b> 3 <b>RAM:</b> 6gb
Video	<b>Memoria:</b> 128MB <b>Controlador:</b> VBOX SVGA	<b>Memoria:</b> 128MB <b>Controlador:</b> VM SVGA	<b>Memoria:</b> 128MB <b>Controlador:</b> VBOX SVGA	<b>Memoria:</b> 128MB <b>Controlador:</b> VBOX SGA

<b>Disco Duro</b>	<b>Formato:</b> vdi <b>Tamaño:</b> 200GB <b>Reserva espacio:</b> Dinámico	<b>Formato:</b> vdi <b>Tamaño:</b> 100GB <b>Reserva espacio:</b> Dinámico	<b>Formato:</b> vdi <b>Tamaño:</b> 100GB <b>Reserva espacio:</b> Dinámico	<b>Formato:</b> vdi <b>Tamaño:</b> 120GB <b>Reserva espacio:</b> Dinámico
-------------------	---	---	---	---

## 2. DIAGRAMA REDE

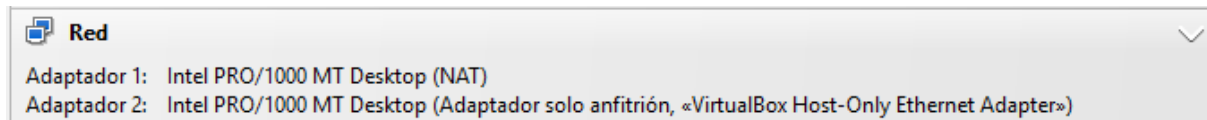


## 4. Ejecución GNU/LINUX

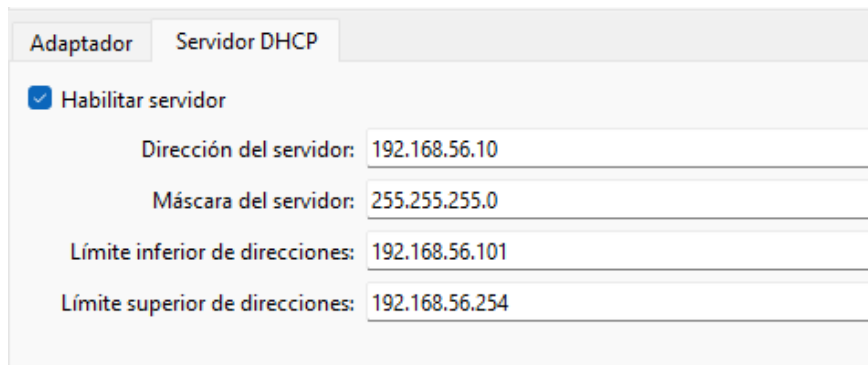
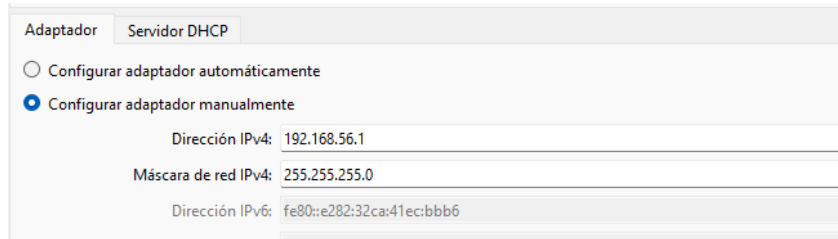
### 3. CREACIÓN DAS MÁQUINAS VIRTUALES

#### CONFIGURACIÓN DE REDE VIRTUAL BOX

→ sería así para as duas maquinas

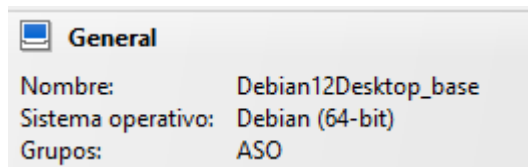


→ configuración solo anfitrión

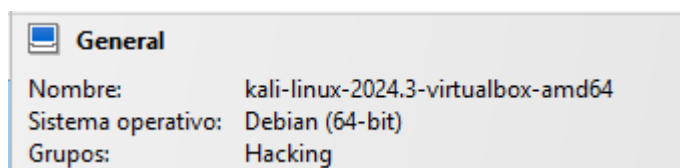


→ para que fije unha ip de forma automática en ese rango  
Añadimos e creamos as máquinas virtuales:

#### 1. Debian 12 Desktop



#### 2. Kali Cliente Atacante



## 4. CONFIGURACIÓN DE REDE DAS MÁQUINAS

### CONFIGURACIÓN DE REDE DO DEBIAN

```

aso@base:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

auto enp0s8
iface amp0s8 inet dhcp

# The loopback network interface
auto lo
iface lo inet loopback

```

ip a

```

aso@base:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c6:24:cc brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 86397sec preferred_lft 86397sec
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ff:99:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 505sec preferred_lft 505sec
    inet6 fe80::a00:27ff:feff:9987/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

```

→ probamos ping a internet:

```

root@base:/home/aso# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=255 time=17.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=255 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=255 time=15.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=255 time=16.0 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4131ms
rtt min/avg/max/mdev = 15.465/16.147/17.437/0.769 ms

```

→ probamos ping a KALI Linux

```
aso@base:~$ ping 192.168.56.100
PING 192.168.56.100 (192.168.56.100) 56(84) bytes of data.
64 bytes from 192.168.56.100: icmp_seq=1 ttl=64 time=0.648 ms
64 bytes from 192.168.56.100: icmp_seq=2 ttl=64 time=0.600 ms
64 bytes from 192.168.56.100: icmp_seq=3 ttl=64 time=0.643 ms
64 bytes from 192.168.56.100: icmp_seq=4 ttl=64 time=0.748 ms
^C
--- 192.168.56.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.600/0.659/0.748/0.054 ms
aso@base:~$
```

## CONFIGURACIÓN DE REDE DO KALI ATACANTE

```
(kali㉿kali)-[~/ssh]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ad:25:87 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 60910sec preferred_lft 60910sec
    inet6 fd00::7c2f:a27c:a17:79dd/64 scope global dynamic noprefixroute
        valid_lft 84852sec preferred_lft 12852sec
    inet6 fe80::7a76:133b:68d0:6e86/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ac:4e:32 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.100/24 brd 192.168.56.255 scope global noprefixroute eth1
        valid_lft forever preferred_lft forever
    inet 192.168.56.101/24 brd 192.168.56.255 scope global secondary dynamic noprefixroute eth1
        valid_lft 588sec preferred_lft 588sec
    inet6 fe80::ef0:3603:8e5f:291d/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

IP NAT: 10.0.2.15

IP Solo Anfitrion: 192.168.56.100 / 192.168.56.101

→ Ping a Debian:

```
(kali㉿kali)-[~/ssh]
└─$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.886 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=0.810 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=1.17 ms
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=0.620 ms
64 bytes from 192.168.56.104: icmp_seq=5 ttl=64 time=0.708 ms
^C
--- 192.168.56.104 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4053ms
rtt min/avg/max/mdev = 0.620/0.839/1.171/0.188 ms
```





→ Como indica o final do anterior comando, é necesario reiniciar para que se produza o proceso de etiquetado

```
/dev/sda1: clean, 211478/1831424 files, 1899968/7323904 blocks
[ OK ] Finished plymouth-read-write.service - Tell Plymouth To Write Out Runtime Data.
Mounting proc-sys-fs-binfmt_misc.mount - Arbitrary Executable File Formats File System...
[ OK ] Mounted proc-sys-fs-binfmt_misc.mount - Arbitrary Executable File Formats File System.
[ OK ] Finished systemd-binfmt.service - Set Up Additional Binary Formats.
[ OK ] Finished systemd-tmpfiles-setup.service - Create System Files and Directories.
Starting systemd-timesyncd.service - Network Time Synchronization...
Starting systemd-update-utmp.service - Record System Boot/Shutdown in UTMP...
[ OK ] Finished systemd-update-utmp.service - Record System Boot/Shutdown in UTMP.
[ OK ] Started systemd-timesyncd.service - Network Time Synchronization.
[ OK ] Reached target sysinit.target - System Initialization.
[ OK ] Reached target time-set.target - System Time Set.
Starting selinux-autorelabel.service - Relabel all filesystems...
Starting alsa-restore.service - Save/Restore Sound Card State...
[ OK ] Finished alsa-restore.service - Save/Restore Sound Card State.
[ OK ] Reached target sound.target - Sound Card.

*** Warning -- SELinux default policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
libsemanage.add_user: user sddm not in password file
Warning: Skipping the following R/O filesystems:
/run/credentials/systemd-sysctl.service
/run/credentials/systemd-sysusers.service
/run/credentials/systemd-tmpfiles-setup-dev.service
/run/credentials/systemd-tmpfiles-setup.service
Relabeling / /home
42.5%
```

→ Unha vez reiniciado e terminado, comprobamos o estado e vemos que xa está activado:

```
aso@base:~$ sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:      /etc/selinux
Loaded policy name:          default
Current mode:                permissive
Mode from config file:      permissive
Policy MLS status:          enabled
Policy deny_unknown status: allowed
Memory protection checking:  actual (secure)
Max kernel policy version:   33
```

→ Para continuar necesitamos cambiar o modo de operación a Enforcing, xa que, o modo permissive como reflexa na captura anterior non bloquea ningunha acción.

```
aso@base:~$ sudo setenforce enforcing
[sudo] password for aso:
```

→ Aquí neste paso tuven un problema, xa que cada vez que executaba o comando anterior, a máquina virtual caía.

→ Revisando os procesos están sendo bloqueados en permissive mode:

```
time->Fri Apr 4 00:27:49 2025
type=PROCTITLE msg=audit(1743719269.468:186): proctitle="/usr/bin/gnome-shell"
type=SYSCALL msg=audit(1743719269.468:186): arch=c000003e syscall=10 success=yes exit=0 a0=315380201000
a1=1000 a2=5 a3=7ffc19d4a080 items=0 ppid=1939 pid=2118 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000
fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=(none) ses=3 comm="gnome-shell" exe="/usr/bin/gnome-shell"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
type=AVC msg=audit(1743719269.468:186): avc: denied { execmem } for pid=2118 comm="gnome-shell" scont
ext=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 tcontext=unconfined_u:unconfined_r:unconfined_
t:s0-s0:c0.c1023 tclass=process permissive=1
```

→ Podemos sacar a conclusión de que SELinux está bloqueando a execución de memoria (execmem) para GNOME Shell. Esto pode facer que a máquina se congele ao cambiar a enforcing

→ Agora imos permitir especificamente que gnome-shell use execmem sin desactivar SELinux:

```
aso@base:~$ sudo ausearch -c "gnome-shell" --raw | audit2allow -M gnome_shell_execmem
***** IMPORTANT *****
To make this policy package active, execute:

semodule -i gnome_shell_execmem.pp
```

→ Con esto busca nos logs as restricións sobre gnome-shell e crea o modulo para permitir a acción

→ Despois pídenos realizar o sudo semodule -i gnome\_shell\_execmem.pp para instalar o módulo de política creado

```
aso@base:~$ sudo semodule -i gnome_shell_execmem.pp
libsemanage.add_user: user sddm not in password file
```

→ Executamolo e dinos que a política estase executando sobre un usuario que non existe no sistema. Imos intentar crear o usuario temporalmente para que SELinux non dea error:

```
aso@base:~$ sudo useradd -r -s /sbin/nologin sddm
```

→ Creamolo e agora imos volver a intentar instalar de novo a política

```
aso@base:~$ sudo semodule -i gnome_shell_execmem.pp
```

→ Agora xa non nos da error e imos comprobar que o módulo está cargado correctamente:

```
aso@base:~$ sudo semodule -l | grep gnome_shell_execmem
gnome_shell_execmem
```

→ Unha vez que aplimos o módulo con éxito, imos volver a probar a cambiar SELinux a enforcing sin ningún fallo.

```
aso@base:~$ sudo setenforce enforcing
[sudo] password for aso:
```

→ Como ben expliquei antes, podemos facer que o enforcing sexa persistente ós reinicios:

```
GNU nano 7.2 /etc/selinux/config *
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# default - equivalent to the old strict and targeted policies
# mls      - Multi-Level Security (for military and educational use)
# src      - Custom policy built from source
SELINUXTYPE=default

# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

→ Agora imos gardar e reiniciar de novo para comprobar que non cambia o modo

```
aso@base:~$ sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:      /etc/selinux
Loaded policy name:           default
Current mode:                 enforcing
Mode from config file:       enforcing
Policy MLS status:           enabled
Policy deny_unknown status:   allowed
Memory protection checking:   actual (secure)
Max kernel policy version:    33
aso@base:~$
```

→ Como podemos ver, xa temos o modo enforcing persistente.

## 6. FORTIFICACIÓN DE SSH CON SELINUX

→ FORTIFICACIÓN:

1. Desactivamos o acceso root por SSH.
2. Obligamos o uso de claves SSH en lugar de contrasinais.
3. Cambiamos o porto por un número alto e rexistrámolo en SELinux.
4. Limitamos os usuarios que poden iniciar sesión.
5. Configuramos un firewall que só permite conexións SSH seguras.
6. Engadimos Fail2Ban para bloquear intentos de acceso sospeitosos.
7. Activamos o MLS de SELinux para un control máis estrito.
8. Monitorizamos eventos en SSH con SELinux e creamos políticas personalizadas.
9. Implementamos 2FA para engadir unha capa extra de seguridade.
10. Implementar Port Knocking para ocultar o porto SSH.
11. Establecer umask para restrinxir permisos por defecto.
12. Configurar SSH Escape Timing para mitigar ataques de forza bruta.
13. Restringir os comandos á executar por SSH
14. Usar SELinux para aplicar un chroot a usuarios SSH
15. Auditoría de comandos con auditd (SELinux-aware)
16. Bloquear conexións TCP de saída con nftables (Firewall de saída)

Este nivel de fortificación fai que SSH sexa **case inexpugnable** nun sistema Linux cunha política SELinux avanzada.

### 1. Restringir o acceso root e usar autenticación por clave pública SSH

→ Accedemos o arquivo de configuración de SSH:

/etc/ssh/sshd\_config

```
aso@base:~$ sudo nano /etc/ssh/sshd_config
[sudo] password for aso:
```

→ Ahora modificamos as opcións:

```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2
#AuthorizedKeysFile .ssh/authorized_keys
# To disable tunneled clear te
PasswordAuthentication no
#PermitEmptyPasswords no
```

→ Con está nova configuración prohibimos o acceso a root e por contraseña e permitimolo por clave pública.

```
aso@base:~$ sudo systemctl restart sshd
```

→ Reiniciamos o servizo SSH para aplicar os cambios

## 2. Cambiar o porto de SSH e protexelo con SELinux

Para evitar ataques automatizados, cambiaremos o porto e configuraremos SELinux para que o permita

→ Volvemos a acceder o arquivo de configuración de SSH:

```
aso@base:~$ sudo nano /etc/ssh/sshd_config
```

```
Include /etc/ssh/sshd_config.d/
```

```
#Port 22
```

```
#AddressFamily any
```

```
#ListenAddress 0.0.0.0
```

```
#ListenAddress ::
```

→ Iremos modificar o porto por defecto de SSH, o 22, por un porto novo, o 2222:

```
Include /etc/ssh/sshd_config.d/*.conf
```

```
Port 2222
```

```
#AddressFamily any
```

```
#ListenAddress 0.0.0.0
```

```
#ListenAddress ::
```

```
#HostKey /etc/ssh/ssh_host_rsa_key
```

```
#HostKey /etc/ssh/ssh_host_ecdsa_key
```

→ Agora temos que configurar SELinux para permitir este porto

```
aso@base:~$ sudo semanage port -a -t ssh_port_t -p tcp 2222
```

→ Comprobamos:

```
aso@base:~$ sudo semanage port -l | grep ssh
```

```
ssh_port_t tcp 2222, 22
```

→ Iremos bloquear o porto 22 para que só funcione co porto 2222:

```
aso@base:~$ sudo semanage port -d -t ssh_port_t -p tcp 22
```

```
ValueError: Port tcp/22 is defined in policy, cannot be deleted
```

→ SELinux non permite eliminar o porto 22 porque está definido na política por defecto.

Iremos bloquear as conexións entrantes por dito porto con **Nftables**:

```
aso@base:~$ sudo systemctl status nftables
• nftables.service - nftables
   Loaded: loaded (/lib/systemd/system/nftables.service; enabled; preset:
   Active: active (exited) since Mon 2025-04-07 12:36:54 CEST; 4s ago
     Docs: man:nft(8)
           http://wiki.nftables.org
   Process: 3872 ExecStart=/usr/sbin/nft -f /etc/nftables.conf (code=exited
   Main PID: 3872 (code=exited, status=0/SUCCESS)
      CPU: 7ms
```

→ Confirmamos que Nftables está activado, ahora imos crear a tabla filter:

```
aso@base:~$ sudo nft add table inet filter
aso@base:~$ sudo nft add chain inet filter input { type filter hook input priority 0 \; }
```

→ Ahora bloqueamos o porto 22:

```
aso@base:~$ sudo nft add rule inet filter input tcp dport 22 drop
```

→ Imos facelo persistente tras reinicios:

```
aso@base:~$ sudo sh -c "nft list ruleset > /etc/nftables.conf"
```

```
aso@base:~$ sudo nft list ruleset
table inet filter {
    chain input {
        type filter hook input priority filter; policy accept;
        tcp dport 22 drop
    }

    chain forward {
        type filter hook forward priority filter; policy accept;
    }

    chain output {
        type filter hook output priority filter; policy accept;
    }
}
```

→ Confirmamos que está creada a regla que bloquea o porto 22.

### 3. Restringir o acceso SSH a usuarios específicos

→ Para que solo certos usuarios poidan conectarse por SSH:

```
aso@base:~$ sudo nano /etc/ssh/sshd_config
```

→ Añadimos a opción:

```
AllowUsers aso martin martin1 martin2
```

→ Unha vez añadida reiniciamos o servicio

### 4. Protexer SSH con Firewall (UFW e NFTABLES)

→ Opción 1, utilizando NFTABLES

Añadimos a regla de permitir o porto 2222

```
aso@base:~$ sudo nft add rule inet filter input tcp dport 2222 accept
```

Verificamos:

```
aso@base:~$ sudo nft list ruleset
table inet filter {
    chain input {
        type filter hook input priority filter;
        tcp dport 22 drop
        tcp dport 2222 accept
    }

    chain forward {
        type filter hook forward priority filte
    }
}
```

→ Opción 2, utilizando UFW

```
aso@base:~$ sudo ufw allow 2222/tcp
Rules updated
Rules updated (v6)
```

Permitimos o novo porto 2222

Comprobamos que se realizaron as modificacións:

```
aso@base:~$ sudo ufw status
Status: active

To Action From
--
2222/tcp ALLOW Anywhere
2222/tcp (v6) ALLOW Anywhere (v6)
```

Ahora imos bloquear o porto 22 con UFW:

```
aso@base:~$ sudo ufw deny 22/tcp
Rule added
Rule added (v6)
```

Comprobamos:

```
aso@base:~$ sudo ufw status
Status: active

To Action From
--
2222/tcp ALLOW Anywhere
22/tcp DENY Anywhere
2222/tcp (v6) ALLOW Anywhere (v6)
22/tcp (v6) DENY Anywhere (v6)
```

## 5. Configurar límites de intentos de conexión con Fail2Ban

Esta medida está centrada sobre todo para evitar ataques de fuerza bruta.

→ Instalamos Fail2Ban:

```
aso@base:~$ sudo apt install fail2ban -y
Lendo as listas de paquetes... Feito
```

→ Creamos unha configuración específica de SSH en Fail2Ban, para eso temos que crear o seguinte arquivo:

```
aso@base:~$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

→ Unha vez creado modificámolo:

→ Añadimos as seguintes regras:

```
[sshd]

# To use more aggressive sshd modes :
# normal (default), ddos, extra or ai
# See "tests/files/logs/sshd" or "fi
#mode    = normal
enabled = true
port = ssh
maxretry = 3
bantime = 600
findtime = 600
logpath = %(sshd_log)s
backend = %(sshd_backend)s
```

→ Poñemos como número máximo de intentos fallidos 3, e poñeremoslle 600 segundos de bloqueo.

→ Comprobamos o estado:

```
aso@base:~$ sudo systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/lib/systemd/system/fail2ban.se
   Active: active (running) since Tue 2025-04-08 0
     Docs: man:fail2ban(1)
  Main PID: 7905 (fail2ban-server)
    Tasks: 5 (limit: 4620)
   Memory: 14.3M
      CPU: 222ms
   CGroup: /system.slice/fail2ban.service
           └─7905 /usr/bin/python3 /usr/bin/fail2b
```

```
Abr 08 00:10:20 base systemd[1]: Started fail2ban.se
Abr 08 00:10:21 base fail2ban-server[7905]: 2025-04-
```

→ Iremos comprobar tamén que Fail2Ban está protexendo o servizo SSH:

```
aso@base:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed:    0
| `-- File list:      /var/log/auth.log
`-- Actions
   |- Currently banned: 0
   |- Total banned:    0
   `-- Banned IP list:
```

→ Como vemos, xa está funcionando, por agora, non hai ningún intento fallido nin IPs baneadas

## 6. Activar o Modo MLS de SELinux para SSH

O Mandatory Access Control (MLS) de SELinux é unha opción avanzada que permite establecer políticas de seguridade máis estritas e granulares, controlando o acceso a recursos do sistema en función do nivel de seguridade asociado a cada proceso e obxecto.



Ao activar este modo para SSH, podemos reforzar a seguridade das conexións SSH, garantindo que só os usuarios autorizados poidan acceder a sistemas sensibles.

→ Comprobamos o arquivo de configuración de SELinux:

```
SELINUX=enforcing
# SELINUXTYPE= can take one of these t
# default - equivalent to the old stri
# mls      - Multi-Level Security (for
# src      - Custom policy built from s
SELINUXTYPE=default
```

→ Modificamos para poñer:

```
# default - equivalent to
# mls      - Multi-Level S
# src      - Custom policy
SELINUXTYPE=mls
```

→ Imos verificar que realmente está activo:

```
aso@base:~$ sudo sestatus | grep "Policy MLS status"
[sudo] password for aso:
Policy MLS status:                enabled
```

## 7. Habilitar SELinux para monitorizar e protexer SSH definindo regras

→ Activamos a primeira regra de auditoría de SSH:

```
aso@base:~$ sudo auditctl -w /etc/ssh/sshd_config -p wa -k ssh_audit
```

→ Agora xa podemos ver os logs de SELinux relacionados con SSH:

```
sudo ausearch -m AVC,USER_AVC -c sshd
```

```
aso@base:~$ sudo ausearch -m AVC,USER_AVC -c sshd
----
time->Fri Apr  4 01:01:43 2025
type=PROCTITLE msg=audit(1743721303.643:162): proctitle=2F7573722F7362696E2F73736864002D74
type=SYSCALL msg=audit(1743721303.643:162): arch=c000003e syscall=138 success=no exit=-13 a0=3 a1=7ffde0
03d990 a2=0 a3=0 items=0 ppid=1 pid=573 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0
fsgid=0 tty=(none) ses=4294967295 comm="sshd" exe="/usr/sbin/sshd" subj=system_u:system_r:sshd_t:s0 key=
(null)
type=AVC msg=audit(1743721303.643:162): avc: denied { getattr } for pid=573 comm="sshd" name="/" dev=
"proc" ino=1 scontext=system_u:system_r:sshd_t:s0 tcontext=system_u:object_r:proc_t:s0 tclass=filesystem
permissive=0
----
time->Fri Apr  4 01:01:43 2025
type=PROCTITLE msg=audit(1743721303.803:174): proctitle=2F7573722F7362696E2F73736864002D44
type=SYSCALL msg=audit(1743721303.803:174): arch=c000003e syscall=138 success=no exit=-13 a0=3 a1=7ffda0
0f2310 a2=0 a3=0 items=0 ppid=1 pid=590 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0
fsgid=0 tty=(none) ses=4294967295 comm="sshd" exe="/usr/sbin/sshd" subj=system_u:system_r:sshd_t:s0 key=
(null)
type=AVC msg=audit(1743721303.803:174): avc: denied { getattr } for pid=590 comm="sshd" name="/" dev=
"proc" ino=1 scontext=system_u:system_r:sshd_t:s0 tcontext=system_u:object_r:proc_t:s0 tclass=filesystem
permissive=0
```

Esta regra é útil para rastrear quen e cando modifica a configuración do SSH, pero non audita a actividade do propio servizo SSH

Para auditar os intentos de login por SSH (incluíndo os intentos fallidos como root), imos añadir regras de auditoría que rastreen eventos relacionados coa autenticación e co proceso sshd.

→ Creación das regras:

**sudo auditctl -w /var/log/secure -p wa -k ssh\_login # Para sistemas que usan este log**  
**sudo auditctl -w /var/log/auth.log -p wa -k ssh\_login**

Estas monitorizan os ficheiros de log de autenticación para escrituras, onde adoitan rexistrarse os intentos de login

**sudo auditctl -a always,exit -F path=/usr/sbin/sshd -F perm=x -k sshd\_exec**

Rastrea a execución do propio sshd

**sudo auditctl -a always,exit -F arch=b64 -F 'auid>=1000' -F 'auid!=4294967295' -S connect -F path=/var/run/sshd.sock -k ssh\_connections**

ou

**sudo auditctl -a always,exit -F arch=b64 -S connect -F 'auid>=1000' -F 'auid!=4294967295' -F path=/var/run/sshd.sock -F key=ssh\_connections**

Rastrea as conexións ao socket de sshd por usuarios reais.

**sudo auditctl -a always,exit -F arch=b64 -S accept -F 'auid>=1000' -F 'auid!=4294967295' -F key=ssh\_access**

Rexistrárase cando o servidor SSH acepte unha nova conexión dun usuario real en sistemas de 64 bits.

**sudo auditctl -a always,exit -F arch=b64 -S fork -F 'auid>=1000' -F 'auid!=4294967295' -F key=ssh\_access**

Registrará a creación de novos procesos por usuarios reais no contexto de SSH en sistemas de 64 bits.

**sudo auditctl -a always,exit -F arch=b64 -S execve -F 'auid>=1000' -F 'auid!=4294967295' -F key=ssh\_access**

Rexistrará a execución de comandos por usuarios reais dentro de sesións SSH en sistemas de 64 bits.

→ Comprobamos que se crearon:

```
aso@base:~$ sudo auditctl -l
-w /var/log/secure -p wa -k ssh_login
-w /var/log/auth.log -p wa -k ssh_login
-w /usr/sbin/sshd -p x -k sshd_exec
-a always,exit -F arch=b64 -S connect -F auid>=1000 -F auid!=-1 -F path=/var/run/sshd.sock -F key=ssh_connections
-a always,exit -F arch=b64 -S accept -F auid>=1000 -F auid!=-1 -F key=ssh_access
-a always,exit -F arch=b64 -S fork -F auid>=1000 -F auid!=-1 -F key=ssh_access
-a always,exit -F arch=b64 -S execve -F auid>=1000 -F auid!=-1 -F key=ssh_access
```

## 8. Configurar autenticación de dous factores (2FA)

→ Para añadir outra capa de seguridade, instalamos google-authenticator:

```
aso@base:~$ sudo apt install libpam-google-authenticator -y
Lendo as listas de paquetes... Feito
Construindo a árbore de dependencias... Feito
Lendo a información do estado... Feito
O seguinte paquete foi instalado automaticamente e xa non é nec
```

→ Unha vez instalado, configurámolo:

```
aso@base:~$ google-authenticator

Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP secret to Google:
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/aso@base%3Fsecret%3DH2U76D
WFYBOI5RYETV5XSTQ3DE%26issuer%3Dbase
```



```
Your new secret key is: H2U76DWFYBOI5RYETV5XSTQ3DE
Enter code from app (-1 to skip):
Code incorrect (correct code 017955). Try again.
Enter code from app (-1 to skip):
Code incorrect (correct code 017955). Try again.
Enter code from app (-1 to skip): 017955
Code confirmed
Your emergency scratch codes are:
67486440
53487906
31676802
79324645
52661403
```

```
Do you want me to update your "/home/aso/.google_authenticator" file? (y/n) y
```

```
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y
```

```
By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
```

```
Do you want to do so? (y/n) y
```

```
If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting? (y/n) y
```

→ Unha vez realizada a configuración inicial de google authenticator, imos activar 2FA no arquivo `/etc/pam.d/ssh`:

```
# to run in the user's context should be run aft
session [success=ok ignore=ignore module_unknowr

# Standard Un*x password updating.
@include common-password

auth required pam_google_authenticator.so
```

→ Unha vez añadido o requerimento de google authenticator, temos que permitir dúas opcións no arquivo `/etc/ssh/sshd_config`:

```
#PermitEmptyPasswords no

ChallengeResponseAuthentication yes
PAMAuthenticationViaKbdInt yes
```

Unha vez terminado, reiniciamos o servizo ssh

## 9. Implementación de Port Knocking

Port Knocking oculta o porto SSH, facéndoo inaccesible ata que se envíe unha secuencia específica de paquetes

→ Instalamos Port Knocking

```
aso@base:~$ sudo apt install knockd
Lendo as listas de paquetes... Feito
Construindo a árbore de dependencias... Feito
Lendo a información do estado... Feito
O seguinte paquete foi instalado automaticamei
libdbus-qlib-1-2
```

→ Unha vez instalado, accedemos o arquivo de configuración `/etc/knockd.conf` e añadimos o novo porto:

```
aso@base:~$ sudo nano /etc/knockd.conf
```

```
GNU nano 7.2 /etc/knockd.conf *
[options]
    UseSyslog

[openSSH]
    sequence      = 7000,8000,9000
    seq_timeout   = 5
    command       = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 2222 -j ACCEPT
    tcpflags      = syn

[closeSSH]
    sequence      = 9000,8000,7000
    seq_timeout   = 5
    command       = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 2222 -j ACCEPT
    tcpflags      = syn
```

→ Como vemos, configurei unha secuencia de portos (7000,8000,9000) para abrir SSH

→ Activamos o servicio Knock:

```
aso@base:~$ sudo systemctl enable knockd --now
Synchronizing state of knockd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable knockd
Created symlink /etc/systemd/system/multi-user.target.wants/knockd.service → /lib/systemd/system/knockd.service.
```

→ Realizamos un status para comprobar o estado

```
aso@base:~$ sudo systemctl status knockd
* knockd.service - Port-Knock Daemon
   Loaded: loaded (/lib/systemd/system/knockd.service; enabled; preset: enabled)
   Active: failed (Result: exit-code) since Tue 2025-04-08 23:49:26 CEST; 1min 14s ago
 Duration: 49ms
    Docs: man:knockd(1)
  Process: 8518 ExecStart=/usr/sbin/knockd $KNOCKD_OPTS (code=exited, status=1/FAILURE)
 Main PID: 8518 (code=exited, status=1/FAILURE)
    CPU: 9ms

Abr 08 23:49:26 base systemd[1]: Started knockd.service - Port-Knock Daemon.
Abr 08 23:49:26 base knockd[8518]: could not open eth0: eth0: No such device exists (No su
Abr 08 23:49:26 base systemd[1]: knockd.service: Main process exited, code=exited, status=
Abr 08 23:49:26 base systemd[1]: knockd.service: Failed with result 'exit-code'.
lines 1-13/13 (END)
```

→ Vemos un error de que knockd non pode atopar a interface de rede eth0

→ Temos que acceder o arquivo /etc/default/knockd, e añadir a interfaz de rede correcta:

```
GNU nano 7.2 /etc/default/knockd
# control if we start knockd at init or not
# 1 = start
# anything else = don't start
# PLEASE EDIT /etc/knockd.conf BEFORE ENABLING
START_KNOCKD=0

# command line options
KNOCKD_OPTS="-i enp0s8"
```

→ Ahora reiniciamos:

```
aso@base:~$ sudo systemctl restart knockd
```

→ Volveremos a comprobar o estado:

```
aso@base:~$ sudo systemctl status knockd
* knockd.service - Port-Knock Daemon
   Loaded: loaded (/lib/systemd/system/knockd.service; enabled; preset: enabled)
   Active: active (running) since Fri 2025-04-25 17:17:08 CEST; 3s ago
     Docs: man:knockd(1)
  Main PID: 17529 (knockd)
    Tasks: 1 (limit: 4620)
  Memory: 616.0K
     CPU: 20ms
   CGroup: /system.slice/knockd.service
           └─17529 /usr/sbin/knockd -i enp0s8

Abr 25 17:17:08 base systemd[1]: Started knockd.service - Port-Knock Daemon.
Abr 25 17:17:08 base knockd[17529]: starting up, listening on enp0s8
```

## 10. Establecer umask para restrinxir permisos

Esto sirve para evitar que os arquivos creados por usuarios SSH teñan permisos inseguros.  
→ Vou añadir umask 077 nos arquivos de configuración do sistema e de Bash para añadir os novos permisos, os cales son moi restrictivos. Solo propietario terá permisos de lectura e escritura

```
aso@base:~$ echo "umask 077" | sudo tee -a /etc/profile /etc/bash.bashrc
umask 077
```

→ Imos comprobar que realmente se añadió ós arquivos:

/etc/profile

```
aso@base:~$ cat /etc/profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1)
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$(id -u)" -eq 0 ]; then
    PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
-

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi
umask 077
```

/etc/bash.bashrc

```
aso@base:~$ cat /etc/bash.bashrc
# System-wide .bashrc file for interactive bash

# To enable the settings / commands in this file
# this file has to be sourced in /etc/profile.

...
return 127
fi
}
fi
umask 077
```

→ Agora imos gardar os cambios realizados nos dous arquivos:

```
aso@base:~$ source /etc/profile
aso@base:~$ source /etc/bash.bashrc
```

→ Comprobamos:

```
aso@base:~$ umask
0077
```

## 11. Configurar SSH Escape Timing

O obxectivo é aumentar a seguridade do servidor SSH desconectando automaticamente as sesións que estiveron activas durante un tempo, para que non poidan ser explotadas.

→ Temos que acceder ó arquivo de configuración do servidor SSH /etc/ssh/sshd\_config:

```
aso@base:~$ sudo nano /etc/ssh/sshd_config
```

→ Unha vez dentro, modificamos as dúas seguintes opcións:

```
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
ClientAliveInterval 300
ClientAliveCountMax 2
#UseDNS no
#PidFile /run/sshd.pid
```

→ O servidor enviará un mensaxe cada 300 segundos, si o cliente non responde, asúmese que a conexión perdiuse. Ademais, si o servidor envía dous mensaxes "keepalive" consecutivos e non recibe resposta, tamén será terminada a conexión.

→ Reiniciamos ssh para gardar os cambios

```
aso@base:~$ sudo systemctl restart ssh
```



## 12. Restringir os comandos á executar por SSH (script personalizado + SELINUX: MLS & RBAC)

### PARTE 1

Podemos limitar os comandos que un usuario pode executar cando inicia sesión por SSH. Isto é útil para usuarios con permisos restringidos, evitando que poidan explorar o sistema ou executar comandos peligrosos.

-Os usuarios que se conecten por SSH só poderán executar comandos de lectura (como ls, cat, less, etc.).

-Non poderán modificar ficheiros, usar nano, rm, mkdir, sudo, nin cargar ficheiros.

→ Primeiro imos crear un grupo de usuarios restrinxidos

```
aso@base:~$ sudo groupadd ssh_readonly
```

→ Asignar os usuarios SSH (que queiramos) a este grupo

```
aso@base:~$ sudo usermod -aG ssh_readonly martin
```

Se queremos añadir outros usuarios como por exemplo Aso, teremos que cambiar martin polo nome do usuario.

→ Imos crear unha shell personalizada para a restricción dos comandos:

```
aso@base:~$ sudo nano /usr/local/bin/ssh_readonly_shell.sh
```

```
GNU nano 7.2 /usr/local/bin/ssh_readonly_shell.sh *
#!/bin/bash

# Lista blanca de comandos permitidos
allowed_cmds="ls|cat|pwd|whoami|uptime|id|date|hostname|df|free|top|less|more|head|tail|uname|env|who|w|man|cal|which"

# Extraemos o comando que intenta usar o usuario
user_cmd=$(echo "$SSH_ORIGINAL_COMMAND" | awk '{print $1}')

# Comprobamos se o comando está na lista blanca
if [[ "$user_cmd" =~ ^($allowed_cmds)$ ]]; then
    $SSH_ORIGINAL_COMMAND
else
    echo "Comando non permitido: $user_cmd"
    exit 1
fi
```

→ Xa expliquei no propio script o que fai cada sección

-Comandos que non se debería añadir para facer unha óptima restricción:

nano, vim, vi, ed, tee, cp, mv, rm → permiten escribir/modificar/eliminar

scp, sftp, wget, curl, rsync → permiten subir/descargar arquivos

tar, zip, unzip → poden escribir ficheiros ao extraer

bash, sh, python, perl, awk, sed → poden abrir shell ou executar scripts

sudo → evidentemente...

→ Dámoslle permisos o script:

```
aso@base:~$ sudo chmod +x /usr/local/bin/ssh_readonly_shell.sh
```

→ Editamos o ficheiro de configuración de ssh:

```
aso@base:~$ sudo nano /etc/ssh/sshd_config
```



→ Accedemos e añadimos:

```
Match Group ssh_readonly
    ForceCommand /usr/local/bin/ssh_readonly_shell.sh
    PermitTTY yes
    AllowTcpForwarding no
    X11Forwarding no
```

→ Esto aplica só aos usuarios do grupo ssh\_readonly e forzará o uso do shell restrinxido.

→ Reiniciamos o servico:

```
aso@base:~$ sudo systemctl restart ssh
```

## PARTE 2 (AVANZADA)

Tamén, para facelo máis avanzado e restrictivo podemos usar SELinux RBAC (Role-Based Access Control) para asignar un rol con permisos reducidos a certos usuarios SSH.

→ Instalamos os paquetes da política MLS:

```
aso@base:~$ sudo apt update
sudo apt install selinux-policy-mls selinux-policy-default
Rcb:1 http://security.debian.org/debian-security bookworm-security InRelease [48,0 kB]
Teño:2 http://deb.debian.org/debian bookworm InRelease
Rcb:3 http://deb.debian.org/debian bookworm-updates InRelease [55,4 kB]
Rcb:4 http://security.debian.org/debian-security bookworm-security/main Sources [152 kB]
Rcb:5 http://security.debian.org/debian-security bookworm-security/main amd64 Packages [7
```

→ Ahora imos realizar un restorecon, para que cada ficheiro e directorio dentro deles, compara o seu contexto de seguridade SELinux actual co contexto que debería ter segundo as políticas SELinux instaladas no sistema. Se hai unha diferenza, restorecon restaura

```
aso@base:~$ sudo restorecon -Rv /etc /home /var
Relabeled /etc/monit from unconfined_u:object_r:etc_t:s0 to unconfined_u:object_r:etc_t:s0
Relabeled /etc/monit/conf-available from unconfined_u:object_r:etc_t:s0 to unconfined_u:object_r:etc_t:s0
Relabeled /etc/monit/monitrc.d from unconfined_u:object_r:etc_t:s0 to unconfined_u:object_r:etc_t:s0
Relabeled /etc/monit/monitrc.d/fail2ban from unconfined_u:object_r:etc_t:s0 to unconfined_u:object_r:etc_t:s0
Relabeled /etc/selinux/mls/contexts from unconfined_u:object_r:selinux_t_r:default_context_t:s0 to unconfined_u:object_r:selinux_t_r:default_context_t:s0
Relabeled /etc/selinux/mls/contexts/virtual_domain_context from unconfined_u:object_r:selinux_t_r:default_context_t:s0 to unconfined_u:object_r:selinux_t_r:default_context_t:s0
```

→ Primeiro, verificamos os roles disponibles con semanage login -l:

```
aso@base:~$ sudo semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Servizo
__default__	user_u	s0-s0	*
root	root	s0-s15:c0.c1023	*
sddm	xdm	s0-s0	*

→ Ahora utilizaremos o usuario martin1

-SELinux inclúe roles como:

**user\_u**: usuario estándar (é o que tes como default)

**staff\_u**: usuario con posibilidade de escalar a root

**guest\_u**: rol moi limitado (lectura básica, non pode instalar, nin subir arquivos)

**xguest\_u**: similar, pero pensado para escritorio

→ Usaremos guest\_u, ideal para SSH restrinxido, imos comprobar os usuarios de SELinux:

```
aso@base:~$ sudo semanage user -l
```

SELinux User	Labeling Prefix	MLS/MCS Level	MLS/MCS Range	SELinux Roles
guest_u	user	s0	s0	user_r
root	sysadm	s0	s0-s15:c0.c1023	auditadm_r secadm_r staff_r sysadm_r
system_r				
staff_u	staff	s0	s0-s15:c0.c1023	auditadm_r secadm_r staff_r sysadm_r
sysadm_u	sysadm	s0	s0-s15:c0.c1023	sysadm_r
system_u	user	s0	s0-s15:c0.c1023	system_r
unconfined_u	unconfined	s0	s0-s15:c0.c1023	system_r unconfined_r
user_u	user	s0	s0	user_r
xdm	user	s0	s0	xdm_r

→ Agora imos proceder a asignar o rol de SELinux user\_u ó usuario martin1:

```
aso@base:~$ sudo semanage login -a -s guest_u martin1
```

→ Verificar a asignación do rol:

```
aso@base:~$ sudo semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Servizo
__default__	user_u	s0-s0	*
martin1	guest_u	s0	*
root	root	s0-s15:c0.c1023	*
sddm	xdm	s0-s0	*

Xa está asignado correctamente o rol guest\_u en SELinux. Terá as restricións de acceso definidas pola política de SELinux para o usuario guest\_u, que adoitan ser moi limitadas para fins de seguridade e illamento.

### 13. Usar SELinux para aplicar un chroot a usuarios SSH

Un chroot é un mecanismo que limita os recursos que pode acceder un usuario. Isto pode ser útil para usuarios que só necesitan acceso a recursos moi específicos e debe impedirse que accedan a outras partes do sistema.

→ Accedemos o arquivo de configuración de SSH, e añadimos a seguinte directiva:

```
# Example of overriding settings on a per-user basis
Match User martin1
    ChrootDirectory /home/martin1
    AllowTcpForwarding no
    ForceCommand /bin/bash
    X11Forwarding no
```

→ A directiva para o usuario martin1:

- Encarcélao no seu directorio persoal (/home/martin1) mediante un "chroot jail", restrinxindo o seu acceso ao sistema de ficheiros.
- Impídelle realizar o tunelado TCP a través da conexión SSH.
- Fórzao a iniciar unha shell Bash interactiva dentro do seu entorno "chroot" ao conectarse.
- Impídelle reenviar aplicacións gráficas X11 a través da conexión SSH.

→ Agora temos que crear os directorios necesarios para que funcione chroot:

```
aso@base:~$ sudo chown root:root /home/martin1
sudo chmod 755 /home/martin1
```

→ O comando /bin/bash ten que existir dentro do chroot

```
aso@base:~$ sudo mkdir -p /home/martin1/bin
sudo cp /bin/bash /home/martin1/bin/
```

→ As bibliotecas necesarias para bash tamén deben estar dentro do chroot

```
aso@base:~$ ldd /bin/bash
linux-vdso.so.1 (0x00007ffce93a3000)
libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007fad2b703000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fad2b522000)
/lib64/ld-linux-x86-64.so.2 (0x00007fad2b88b000)
```

→ Para eso imos generar un script automatizado:

```
GNU nano 7.2                                instalar_bash.sh
#!/bin/bash

CHROOT_DIR="/home/martin1"

echo "[+] Copiando /bin/bash a $CHROOT_DIR/bin"
mkdir -p "$CHROOT_DIR/bin"
cp /bin/bash "$CHROOT_DIR/bin"

echo "[+] Analizando dependencias de bash..."
libs=$(ldd /bin/bash | awk '{print $3}' | grep "^/")

echo "[+] Copiando librerías necesarias..."
for lib in $libs; do
    dest_dir="$CHROOT_DIR$(dirname $lib)"
    mkdir -p "$dest_dir"
    cp "$lib" "$dest_dir"
    echo "    Copiado: $lib -> $dest_dir"
done

echo "[+] Copia completada. Bash e dependencias están no entorno chroot."

# Verificación rápida
echo -e "\n[+] Verifica que existe:"
ls -l "$CHROOT_DIR/bin/bash"
```

→ Este script automatiza os pasos necesarios para crear un entorno chroot mínimo para o usuario martin1 que inclúe o shell Bash e as súas librerías esenciais. Este é un paso crucial para que a directiva ChrootDirectory e ForceCommand /bin/bash no ficheiro de configuración do SSH funcionen correctamente, permitindo que o usuario martin1 teña unha shell interactiva pero restrinxida ao seu directorio persoal.

→ Damoslle permisos:

```
aso@base:~$ sudo chmod +x instalar_bash.sh
```

→ Executámolo:

```
aso@base:~$ sudo ./instalar_bash.sh
[+] Copiando /bin/bash a /home/martin1/bin
[+] Analizando dependencias de bash...
[+] Copiando librerías necesarias...
    Copiado: /lib/x86_64-linux-gnu/libtinfo.so.6 → /home/martin1/lib/x86_64-linux-gnu
    Copiado: /lib/x86_64-linux-gnu/libc.so.6 → /home/martin1/lib/x86_64-linux-gnu
[+] Copia completada. Bash e dependencias están no entorno chroot.

[+] Verifica que existe:
-rwx-----. 1 root root 1265648 Abr 14 17:39 /home/martin1/bin/bash
```

→ Tras esto reiniciamos SSH

```
aso@base:~$ sudo systemctl restart ssh
```

#### 14. Auditoría de comandos con auditd (SELinux-aware)

Instalar e configurar o sistema de auditoría de Linux auditd, que rexistra os comandos executados polos usuarios, especialmente os que afectan ao sistema, e se integra perfectamente con SELinux. Esto permite monitorizar e ter traza de eventos críticos mesmo cando SELinux non bloquea, pero si xenera avisos.

→ Instalamos:

```
aso@base:~$ sudo apt install auditd audispd-plugins
Lendo as listas de paquetes... Feito
Construindo a árbore de dependencias... Feito
Lendo a información do estado... Feito
auditd is already the newest version (1:3.0.9-1).
```

→ Comprobamos o UID de martin1:

```
aso@base:~$ id martin1
uid=1002(martin1) gid=1003(martin1) grupos=1003(martin1)
```

→ Activar auditoría de execución de comandos por usuarios (monitorizar martin1)

```
aso@base:~$ sudo auditctl -a always,exit -F arch=b64 -F euid=1002 -S execve -k comandos_martin1
```

-Este comando configura o sistema de auditoría para:

- Sempre (always) rexistre a saída (exit) de calquera chamada ao sistema execve.
- Só se aplique a procesos que se executen en arquitectura de 64 bits (arch=b64).
- Estean sendo executados polo usuario cun Effective User ID de 1002.
- Todos os rexistros generados por esta regra sexan etiquetados ca clave

comandos\_martin1

→ Aquí é donde podemos comprobar os rexistros unha vez o usuario os introduzca:

```
aso@base:~$ sudo ausearch -k comandos_martin1
----
time->Tue Apr 15 11:14:03 2025
type=PROCTITLE msg=audit(1744708443.518:9419): proctitle=617564697463746C002D6100616C776179732C657869740
02D46006172636800623634002D4600657569640031303032002D5300657865637665002D6B00636F6D616E646F735F6D6172746
96E31
type=SOCKADDR msg=audit(1744708443.518:9419): saddr=1000000000000000000000000000000000
type=SYSCALL msg=audit(1744708443.518:9419): arch=c000003e syscall=44 success=yes exit=1072 a0=4 a1=7ffd
fb74f9b0 a2=430 a3=0 items=0 ppid=13019 pid=13020 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgi
d=0 fsgid=0 tty=pts1 ses=3 comm="auditctl" exe="/usr/sbin/auditctl" subj=unconfined_u:unconfined_r:audit
ctl_t:s0-s0:c0.c1023 key=(null)
type=CONFIG_CHANGE msg=audit(1744708443.518:9419): auid=1000 ses=3 subj=unconfined_u:unconfined_r:auditc
tl_t:s0-s0:c0.c1023 op=add_rule key="comandos_martin1" list=4 res=1
```

→ Imos facelo persistente tras reinicio, para eso temos que acceder a o ficheiro de configuración de regras persistentes de auditd:

```
aso@base:~$ sudo nano /etc/audit/rules.d/comandos_martin1.rules
```

→ Añadimos a regra:

```
GNU nano 7.2 /etc/audit/rules.d/comandos_martin1.rules
-a always,exit -F arch=b64 -F euid=1002 -S execve -k comandos_martin1
```

→ Verificamos que a regra está activa:

```
aso@base:~$ sudo auditctl -l
-a always,exit -F arch=b64 -S execve -F euid=1002 -F key=comandos_martin1
```

A regra está cargada no sistema e registrará os comandos executados polo usuario martin1

Esto sería para poder revisar os comandos de martin1 introducidos como usuario no servidor debian. Para poder auditar o usuario martin1 na conexión SSH desde un cliente sería así:

→ Accedemos o arquivo de regras de audit e añadimos:

```
GNU nano 7.2 /etc/audit/rules.d/audit.rules *
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 60000

## Set failure mode to syslog
-f 1

# Regras para sshd (por defecto)
-w /var/log/secure -p wa -k ssh_login
-w /var/log/auth.log -p wa -k ssh_login
-a always,exit -F path=/usr/sbin/sshd -F perm=x -k sshd_exec
-a always,exit -F arch=b64 -S connect -F auid>=1000 -F auid!=4294967295 -F path=/var/run/sshd.sock -k s
-a always,exit -F arch=b64 -S accept -F auid>=1000 -F auid!=4294967295 -F key=ssh_access
-a always,exit -F arch=b64 -S fork -F auid>=1000 -F auid!=4294967295 -F key=ssh_access

# Regra ESPECÍFICA para martin1
-a always,exit -F arch=b64 -F uid=1002 -S execve -k comandos_martin1

# Regra xeral para execve doutros usuarios (se é necesario)
-a always,exit -F arch=b64 -S execve -F auid>=1000 -F auid!=4294967295 -k execve_otros
```

**# Regra ESPECÍFICA para martin1**

*-a always,exit -F arch=b64 -F uid=1002 -S execve -k comandos\_martin1*

Audita a execución de programas (64 bits) feita polo usuario martin1 (UID 1002), etiquetando os eventos como comandos\_martin1.

**# Regra xeral para execve doutros usuarios (se é necesario)**

*-a always,exit -F arch=b64 -S execve -F auid>=1000 -F auid!=4294967295 -k execve\_otros*

Audita a execución de programas (64 bits) feita por usuarios reais que iniciaron sesión (baseándose no seu ID de auditoría), etiquetando os eventos como execve\_otros

## 15. Bloquear conexións TCP de saída con nftables (Firewall de saída)

Con nftables podes bloquear as conexións TCP de saída, por exemplo, permitindo só as necesarias (como SSH) e bloqueando o resto. Limita a capacidade dun proceso ou usuario de establecer conexións externas non autorizadas. Fundamental en entornos de alto control como servidores compartidos ou usuarios invitados.

→ Polo tanto, imos crear a regra nftables que realice o anterior:

```
aso@base:~$ sudo nft add rule inet filter output meta skuid martin tcp dport != 2222 drop
```

A regra fai o seguinte; bloquea (descarta) todo o tráfico TCP saínte generado polo usuario martin1, excepto aquel que teña como porto de destino o 2222.

→ Comprobamos que realmente foi añadida:

```
aso@base:~$ sudo nft list ruleset
```

```
chain output {
    type filter hook output priority filter; policy accept;
    meta skuid 1002 tcp dport != 2222 drop
}
```

→ Uha vez comprobado, reiniciamos Nftables

```
aso@base:~$ sudo systemctl restart nftables
```

Tamén se consideraron máis opcións de fortificación de SSH con SELinux, como por exemplo, habilitar o booleano `deny_tcp_connect` para impedir conexións TCP saíntes, pero esta opción non está dispoñible na política SELinux actual. No seu lugar, restrinxíronse os permisos de rede mediante o uso do usuario SELinux `guest_u`, que non permite a apertura de sockets, e pódese complementar cunha regra iptables para evitar tráfico saínte por parte de usuarios restrinxidos.

## 7. PROBAS DA FORTIFICACIÓN DE SSH

Primeiro de todo, comprobamos que se conecta a máquina co cliente kali:

→ De debian a kali:

```
aso@base:~$ ping 192.168.56.100
PING 192.168.56.100 (192.168.56.100) 56(84) bytes of data.
64 bytes from 192.168.56.100: icmp_seq=1 ttl=64 time=1.60 ms
64 bytes from 192.168.56.100: icmp_seq=2 ttl=64 time=0.727 ms
64 bytes from 192.168.56.100: icmp_seq=3 ttl=64 time=0.953 ms
^C
--- 192.168.56.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.727/1.093/1.601/0.370 ms
```

→ De kali a debian:

```
(kali㉿kali)-[~/Desktop]
$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.882 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=4.65 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=3.23 ms
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=5.77 ms
64 bytes from 192.168.56.104: icmp_seq=5 ttl=64 time=3.74 ms
64 bytes from 192.168.56.104: icmp_seq=6 ttl=64 time=3.22 ms
^C
--- 192.168.56.104 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 0.882/3.580/5.765/1.497 ms
```



→ Desde o Kali tamén podemos realizar un escaneado de nmap para comprobar que detecta a máquina:

```
(kali@kali)-[~/Desktop]
$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.882 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=4.65 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=3.23 ms
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=5.77 ms
64 bytes from 192.168.56.104: icmp_seq=5 ttl=64 time=3.74 ms
64 bytes from 192.168.56.104: icmp_seq=6 ttl=64 time=3.22 ms
^C
--- 192.168.56.104 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 0.882/3.580/5.765/1.497 ms
```

Como podemos ver, detecta a máquina .104

→ Agora imos comprobar que portos ten abertos a debian:

```
(kali@kali)-[~/Desktop]
$ sudo nmap -n -Pn -sV -A -O -p- 192.168.56.104
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-24 16:20 CEST
Nmap scan report for 192.168.56.104
Host is up (0.00084s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    filtered ssh
2222/tcp  open  ssh      OpenSSH 9.2p1 Debian 2+deb12u5 (protocol 2.0)
| ssh-hostkey:
|   256 28:80:5f:25:e8:aa:42:5c:2d:fc:f6:88:5d:4f:ef:a8 (ECDSA)
|_  256 b6:78:c8:5f:67:f8:e7:ab:22:87:5a:bc:48:64:ba:71 (ED25519)
MAC Address: 08:00:27:FF:99:87 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT ADDRESS
1 0.83 ms 192.168.56.104

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.76 seconds
```

Como podemos observar no escaneo:

- O porto 22 ssh filtered, significa que o porto 22 non responde ao escaneo polo que está bloqueado ou cancelado. Polo que confirmamos que a fortificación está funcionando
- O porto 2222 ssh open, que foi o porto que añadimos nas regras de SELinux

## AGORA IMOS COMENZAR A PROBA DAS REGLAS:

### 1. Desactivamos o acceso root por SSH.

[vídeo proba acceso root](#)

Para verificar a correcta implementación da directiva de seguridade que desactiva o acceso root por SSH, realizouse un intento de conexión ao servidor utilizando o usuario root a través do porto non estándar configurado (2222). Aínda que o cliente SSH solicitou un



contrasinal, a análise dos logs do servidor SSH revelou entradas de 'fallo de autenticación' (authentication failure) para o usuario root, confirmando que o acceso directo(aínda introducindo ben a contraseña de root) está prohibido.

No propio vídeo podemos comprobar que o acceso a root no servidor SSH sí o podemos realizar, polo que se pode comprobar que a contraseña é a correcta e funciona.

## 2. Obrigamos o uso de claves SSH en lugar de contrasinais.

### [video rechazo por claves SSH](#)

Para verificar que o sistema obriga o uso de claves SSH e non permite a autenticación baseada en contrasinal, realizouse unha proba desde a máquina cliente (Kali Linux) sen proporcionar unha clave privada válida para o usuario aso no servidor (conectando ao porto non estándar 2222).

Ahora imos intentar acceder como accedería un usuario permitido:

→ Generamos a clave

```
(kali@kali)-[~/Desktop]
$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Enter passphrase for "/home/kali/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:CRujU/V3KAyrwGnUG/yJMkkKVeM5iBesgFnDwOjQ/j8 kali@kali
The key's randomart image is:
+----[RSA 4096]-----+
|O=.+o  o|
|Bo+B.++. = .|
|+=+ooO.==..+ o .|
|.oo.+=o*o. o .|
|.ooo S|
|..|
|.E|
|. |
+----[SHA256]-----+
```

→ Comprobamos:

```
(kali@kali)-[~/ssh]
$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

→ Copiamos a clave

```
(kali@kali)-[~/ssh]
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACvFL6/N0Whjx0GrthBs/V2KnnkICiFDvWcJdhgU8Mj1wS09rK2ApX/0NZBpneTc2k+75+2miP++qv3
z8ixV8voERN/Ji+Hakrklke0cp+tslFCOp0S1zHfgiKwLTgxSNwSLWxs182sVKfvJBtuEjcYtmMyMLEMeZKIHERrt/GuUxMnd6ZIKa/DDnu5h+lpZLzk
z4s65N/M21bD/GN+u1d00fivqrCwnJPqup2519LjbxysUKakgPAYVgVn0tCXy3/BrdCnJamr6u2uNmBd/x8TAAem5r1zBA9Pg4ngmKZjzZCKUKA2VFK
RFF+pkkWhyrKvGISFe+qkErLWJoJDM/mQir48ii17q7ABg2Ns3JAw712YX50W/bm85Iqz87qxtJ0FH5FkuCL3eJDviwrPGurWdv+xEIDLbVl+HtyCmw
Y/bhtl9jKgL9dmh7EotQSjfkLwto0dv6MZOYLUnuU9A9xyQaEM8++4dxF4SNnTB2nxjtRIh0GbD340W42f9EfFar/UwogxmIWuYbHE/p5oxhs1l/XnEX
45slb2/iUI689n1f8GD2XbMpyLK+wz4XaGyuhbRtrRDLmZ3PPLm7fIla2azlpaZkKWiW0Dp/c08mDfSGK+gY4qaG2MNkzgTgiPuhv19jijznzUoIgtte
Pwiwji/EoKJSnQYh6h0BnF8e6w== kali@kali
```

→ Imos o servidor de ssh e pegamos a clave publica en:

```
aso@base:~/ssh$ sudo nano authorized_keys
```

→ Unha vez pegada, volvemos o kali e probamos:

[video clave SSH Sí](#)

### 3. Cambiamos o porto por un número alto e rexistrámolo en SELinux.

[video porto 22 no 2222 si](#)

Como podemos ver, o intento de conexión mediante o porto 22, que é o porto de SSH por defecto, non se pode levar a cabo. No obstante, a conexión polo novo porto sí se pode levar a cabo.

### 4. Limitamos os usuarios que poden iniciar sesión.

[video límite usuarios SSH](#)

→ Configuración anterior no arquivo de configuración de ssh:

```
AllowUsers aso martin martin1 martin2
```

-Ao intentar conectar cun usuario incluído na lista AllowUsers (martin1), a conexión foi denegada cun Permission denied (publickey). A análise dos logs do servidor non mostrou un erro de usuario inválido para este intento, o que suxire que a conexión foi permitida a nivel de usuario, pero fallou a autenticación debido á falta dunha clave pública válida desde o cliente.

-Ao intentar conectar cun usuario non incluído na lista AllowUsers (pepito), a conexión tamén resultou nun Permission denied (publickey) no cliente. Sen embargo, a análise dos logs do servidor revelou a mensaxe 'Invalid user pepito'. Isto confirma que a directiva AllowUsers está a funcionar para rechazar usuarios non permitidos nunha fase temperá da conexión, antes de intentar a autenticación.

### 5. Configuramos un firewall que só permite conexións SSH seguras.

```
(kali@kali)~[/Desktop]
$ sudo nmap -n -Pn -sV -A -O -p- 192.168.56.104
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-24 16:20 CEST
Nmap scan report for 192.168.56.104
Host is up (0.00084s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    filtered ssh
2222/tcp  open  ssh      OpenSSH 9.2p1 Debian 2+deb12u5 (protocol 2.0)
```

Na captura que vimos anteriormente:

-Porto 22 filtered: Isto indica que o firewall está a bloquear activamente as conexións entrantes ao porto estándar de SSH (porto 22).

-Porto 2222 open: Este é o teu porto SSH non estándar que configurei, e o estado open indica que nmap puido establecer unha conexión TCP con éxito nese porto, confirmando que o servizo SSH está escoitando correctamente no porto 2222 e que o firewall permite as conexións a ese porto.

### 6. Engadimos Fail2Ban para bloquear intentos de acceso sospeitosos.

[video Fail2Ban](#)

Para verificar a funcionalidade de Fail2Ban na protección do servizo SSH, realizouse unha proba simulando intentos de acceso sospeitosos desde a máquina cliente (Kali Linux). O jail de SSH configurouse cun límite de intentos fallidos (maxretry) dentro do período de tempo

predeterminado (findtime), tras o cal a dirección IP do atacante sería bloqueada durante 600 segundos (bantime = 600).

## 7. Activamos o MLS de SELinux para un control máis estrito.

```
aso@base:~$ sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:       /etc/selinux
Loaded policy name:            mls
Current mode:                  enforcing
Mode from config file:         enforcing
Policy MLS status:             enabled
Policy deny_unknown status:    denied
Memory protection checking:    actual (secure)
Max kernel policy version:    33
```

A imaxe superior mostra o estado de SELinux no servidor despois de ser configurado e reiniciado. A saída do comando sestatus indica que SELinux está enabled e a política de seguridade mls (Multi-Level Security) está cargada. O Current mode é enforcing, o que significa que SELinux está aplicando activamente as políticas de seguridade, denegando as accións que non están explicitamente permitidas. O Mode from config file tamén mostra enforcing, confirmando que esta configuración persistirá nos reinicios futuros.

## 8. Monitorizamos eventos en SSH con SELinux e creamos políticas personalizadas.

[vídeo Monitorización con SELinux](#)

Este log de auditoría de SELinux demostra claramente que o intento de inicio de sesión como root a través de SSH foi rexistrado polo sistema de auditoría. O campo res=failed confirma que a autenticación fallou, como se esperaba debido á configuración de PermitRootLogin no en sshd\_config

## 9. Implementamos 2FA para engadir unha capa extra de seguridade.

[vídeo 2FA](#)

Para verificar a correcta implementación da autenticación de dous factores (2FA) utilizando Google Authenticator, realizouse un intento de inicio de sesión por SSH ao servidor (porto 2222) co usuario aso. Para esta proba específica, deshabilitouse temporalmente a autenticación por claves SSH para permitir o acceso mediante contraseña (temporalmente activado) seguido da solicitude do código de verificación 2FA.

Como se pode comprobar no vídeo, o sistema solicitou un 'Verification code'. Este comportamento confirma que o segundo factor de autenticación está activo e funcionando.

## 10. Implementar Port Knocking para ocultar o porto SSH.

[vídeo Port Knocking](#)

Implementei a técnica de port knocking co obxectivo de engadir unha capa adicional de seguridade ao ocultar o porto de acceso SSH. Deste xeito, o porto permanece pechado e invisible a escaneos de red hasta que se envía unha secuencia predefinida a outros portos. Para verificar a correcta funcionalidade desta implementación, enviei desde a máquina cliente Kali a secuencia de portos configurada (7000, 8000, 9000) ao servidor Debian. Tras enviar a secuencia correcta, comprobei mediante a ferramenta netcat (nc -zv) que o porto 2222 se abriu temporalmente, permitindo así unha conexión SSH

### 11. Establecer umask para restringir permisos por defecto.

[video UMASK](#)

Co obxectivo de prever a creación de arquivos con permisos inseguros por parte dos usuarios que acceden mediante SSH.

Para verificar a correcta aplicación desta configuración accedín por SSH. A saída do comando umask confirmou a activación da máscara co valor 077. Posteriormente, creei un arquivo de proba. Ó inspeccionar os permisos do arquivo creado co comando ls -l, observouse que os permisos resultantes eran -rw----- (600).

Desta forma, asegúrase que todos os arquivos e directorios creados por usuarios SSH teñan por defecto permisos restritivos, reforzando a seguridade do sistema.

### 12. Configurar SSH Escape Timing para mitigar ataques de forza bruta.


A desconexión automática non require intervención do usuario nin probas complexas máis aló de verificar que, tras un período de inactividade establecido, a sesión finaliza correctamente, liberando recursos e minimizando riscos potenciais.

### 13. Restringir os comandos á executar por SSH

[video Restriccion Comandos](#)

A restrición de comandos básica para o usuario aso mediante un script e ForceCommand está funcionando correctamente. Os comandos da lista branca execútanse, e os comandos non permitidos son bloqueados cunha mensaxe

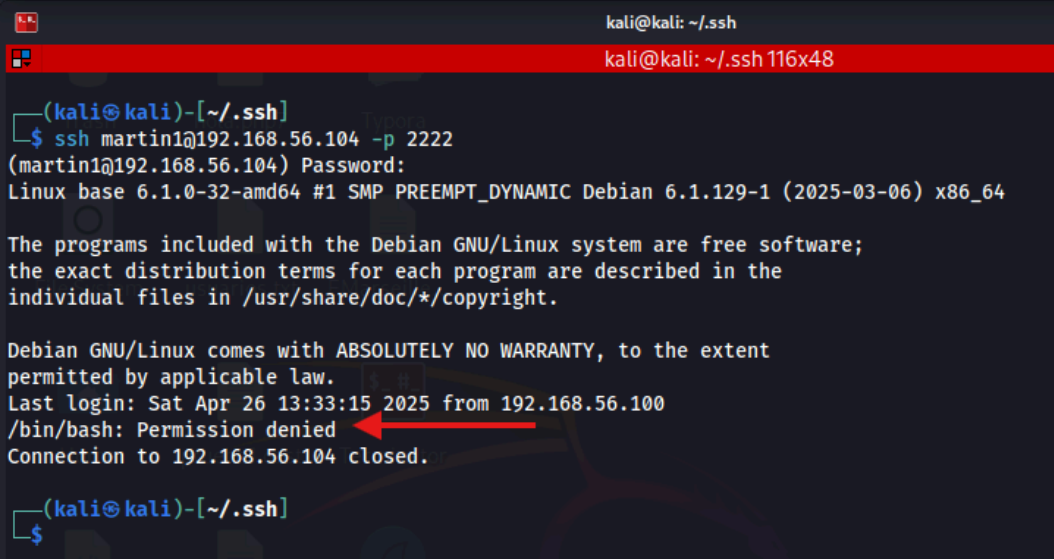
### 14. Usar SELinux para aplicar un chroot a usuarios SSH



```
aso@base: ~  
aso@base:~$ sudo chroot /home/martin1 /bin/bash  
bash-5.2# pwd  
/  
bash-5.2# ls  
bin home lib lib64  
bash-5.2# cd /bin  
cd: cannot access '/bin'  
bash-5.2# ls /lib  
ls: cannot access '/lib'  
bash-5.2#
```

Unha vez dentro do ambiente chroot configurado para /home/martin1, o usuario non pode navegar fóra deste directorio raíz virtual. Os intentos de usar cd .. para subir un nivel e de acceder ao directorio /etc (que está fóra do ambiente chroot no sistema real) fallan. Isto confirma que o confinamento do directorio raíz mediante chroot está funcionando como se esperaba. O usuario martin1, ao operar dentro deste ambiente, estará restrinxido ao

sistema de ficheiros contido en /home/martin1



```
kali@kali: ~/.ssh
kali@kali: ~/.ssh 116x48

(kali@kali)~[~/.ssh]
$ ssh martin1@192.168.56.104 -p 2222
(martin1@192.168.56.104) Password:
Linux base 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr 26 13:33:15 2025 from 192.168.56.100
/bin/bash: Permission denied
Connection to 192.168.56.104 closed.

(kali@kali)~[~/.ssh]
$
```

O fallo que aínda impide a conexión exitosa desde o cliente Kali unha configurado o Chroot (e, polo tanto, a proba real do chroot a través de SSH) probablemente reside nunha combinación de factores relacionados cos permisos e a configuración específica de SSH para o ambiente chroot.

Para que esto funcionase, creei a estrutura de directorios bin dentro do chroot e copiei o shell bash e as súas bibliotecas necesarias. Encontrei e solucionei problemas de permisos no directorio /home/martin1/bin que impedían o acceso. Finalmente, probei o confinamento do chroot localmente no servidor usando o comando chroot, verificando que a navegación fóra de /home/martin1 (aparecendo como / dentro do chroot) estaba restrinxida.

## 15. Auditoría de comandos con auditd (SELinux-aware)

### [Video Auditoria de Comandos](#)

Esta auditoría permítenos ter un rexistro detallado de todas as accións realizadas polo usuario martin1 durante a súa sesión SSH, o que é crucial para a seguridade e a monitorización do sistema. En caso de incidentes ou comportamento sospeitoso, podemos revisar estes logs para entender as accións realizadas polo usuario

## 16. Bloquear conexións TCP de saída con nftables (Firewall de saída)

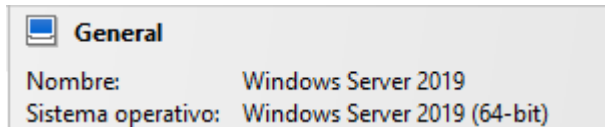
### [video NFTables](#)

Se observas os resultados esperados en cada proba, confirmarás que o firewall de saída con nftables está funcionando correctamente para o usuario martin, bloqueando todas as conexións TCP menos o porto 2222. O tráfico non TCP (como ICMP) non se verá afectado por esta regra específica.

## 5. Ejecución WINDOWS

### 1. CREACIÓN DAS MÁQUINAS VIRTUALES

Windows Server 2019

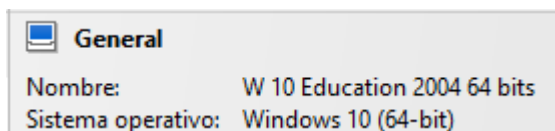


**General**

Nombre: Windows Server 2019

Sistema operativo: Windows Server 2019 (64-bit)

Windows 10 Cliente



**General**

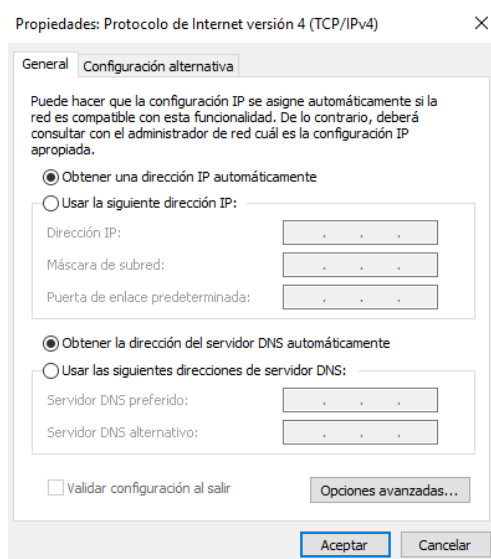
Nombre: W 10 Education 2004 64 bits

Sistema operativo: Windows 10 (64-bit)

### 2. CONFIGURACIÓN DE REDE DAS MÁQUINAS

#### CONFIGURACIÓN DE REDE DO WINDOWS SERVER

→ adaptador nat:



Propiedades: Protocolo de Internet versión 4 (TCP/IPv4) X

General Configuración alternativa

Puede hacer que la configuración IP se asigne automáticamente si la red es compatible con esta funcionalidad. De lo contrario, deberá consultar con el administrador de red cuál es la configuración IP apropiada.

☒ Obtener una dirección IP automáticamente

☐ Usar la siguiente dirección IP:

Dirección IP: . . .

Máscara de subred: . . .

Puerta de enlace predeterminada: . . .

☒ Obtener la dirección del servidor DNS automáticamente

☐ Usar las siguientes direcciones de servidor DNS:

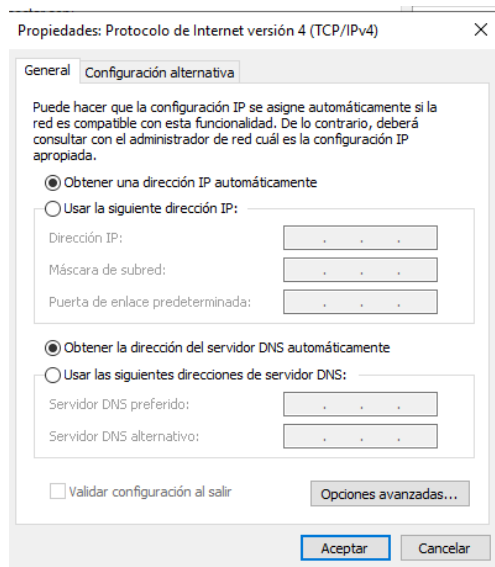
Servidor DNS preferido: . . .

Servidor DNS alternativo: . . .

☐ Validar configuración al salir Opciones avanzadas...

Aceptar Cancelar

→ adaptador solo anfitrion:



Deixamos os dous en DHCP xa que a NAT dame igual a ip que adapte, e o adaptador solo anfitrión xa lle puxen os intervalos na configuración do virtual box.

→ Comprobamos configuración ip:

```
C:\Users\Administrador>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

    Sufixo DNS específico para la conexión. . . :
    Dirección IPv6 . . . . . : fd00::f0ae:f6df:7f0c:e2c5
    Vínculo: dirección IPv6 local. . . : fe80::f0ae:f6df:7f0c:e2c5%6
    Dirección IPv4. . . . . : 10.0.2.15
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . : fe80::2%6
                                           10.0.2.2

Adaptador de Ethernet Ethernet 2:

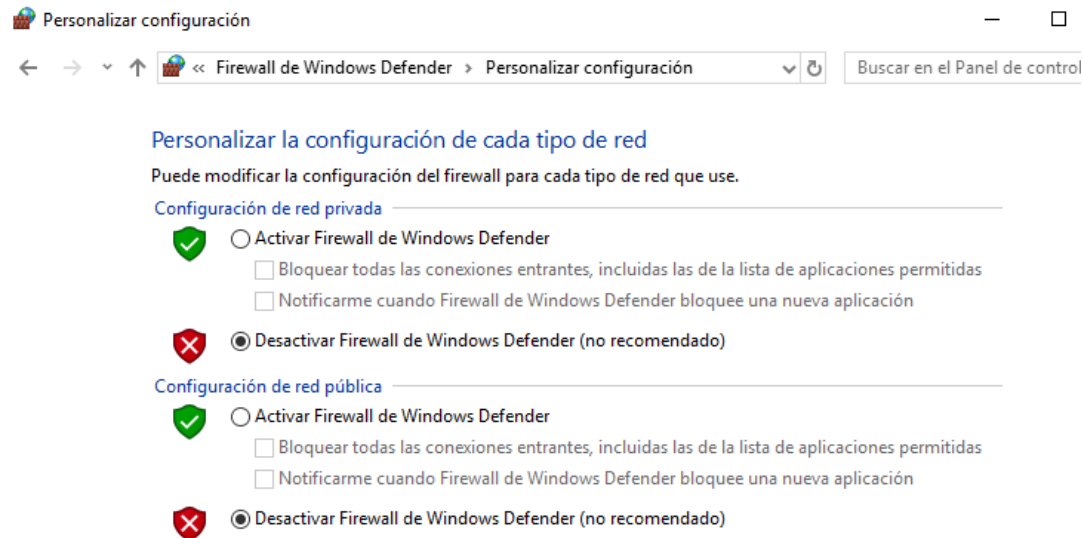
    Sufixo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::5913:564d:205b:69d1%4
    Dirección IPv4. . . . . : 192.168.56.105
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . :
```

Nat: 10.0.2.15

Solo Anfitrión: 192.168.56.105



→ En windows debemos desactivar o Firewall de Windows Defender para que podamos recibir conexión de entrada e de saída:



Se o deixamos por defecto, estas conexións serán automáticamente rechazadas

→ ping a google:

```
C:\Users\Administrador>ping 8.8.8.8

Haciendo ping a 8.8.8.8 con 32 bytes de datos:
Respuesta desde 8.8.8.8: bytes=32 tiempo=15ms TTL=255
Respuesta desde 8.8.8.8: bytes=32 tiempo=15ms TTL=255
Respuesta desde 8.8.8.8: bytes=32 tiempo=16ms TTL=255
Respuesta desde 8.8.8.8: bytes=32 tiempo=15ms TTL=255

Estadísticas de ping para 8.8.8.8:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 15ms, Máximo = 16ms, Media = 15ms
```

→ ping a debian:

```
C:\Users\Administrador>ping 192.168.56.104

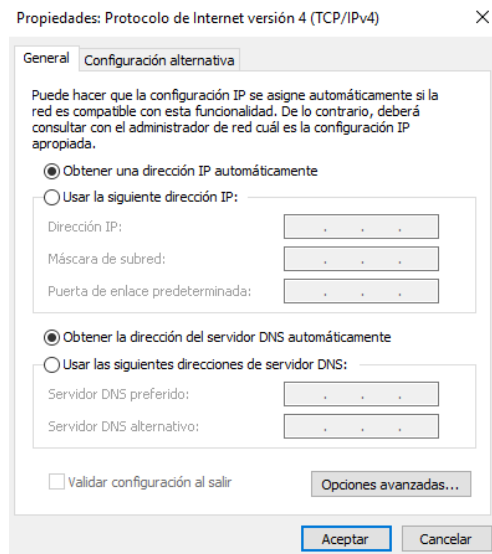
Haciendo ping a 192.168.56.104 con 32 bytes de datos:
Respuesta desde 192.168.56.104: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.56.104: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.56.104: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.56.104: bytes=32 tiempo<1m TTL=64

Estadísticas de ping para 192.168.56.104:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 0ms, Máximo = 1ms, Media = 0ms
```

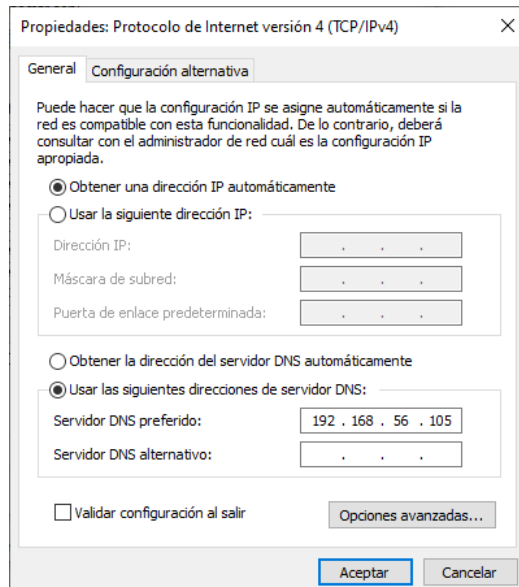


## CONFIGURACIÓN DE REDE DO WINDOWS 10 CLIENTE

→ adaptador nat:



→ adaptador solo anfitrión:



Igual que o Windows Server, deixamos os dous en DHCP xa que a NAT dame igual a ip que adapte, e o adaptador solo anfitrión xa lle puxen os intervalos na configuración do virtual box. En este caso, en DNS debemos poñer a IP do Windows Server.

Como no Win Server, tamén desactivamos o firewall para evitar bloqueo no momento de realizar conexións.

→ Configuración:

```
C:\Windows\system32>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

    Sufijo DNS específico para la conexión. . . :
    Dirección IPv6 . . . . . : fd00::b190:3739:45be:e546
    Dirección IPv6 temporal. . . . . : fd00::e5b0:dbc1:6697:73cc
    Vínculo: dirección IPv6 local. . . : fe80::b190:3739:45be:e546%6
    Dirección IPv4. . . . . : 10.0.2.15
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . : fe80::2%6
                                                10.0.2.2

Adaptador de Ethernet Ethernet 2:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::cc86:9ce0:aa4b:8ccf%13
    Dirección IPv4. . . . . : 192.168.56.106
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . . :

C:\Windows\system32>
```

→ Ping a Windows Server:

```
C:\Windows\system32>ping 192.168.56.105

Haciendo ping a 192.168.56.105 con 32 bytes de datos:
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo=1ms TTL=128

Estadísticas de ping para 192.168.56.105:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 0ms, Máximo = 1ms, Media = 0ms

C:\Windows\system32>_
```

→ Ping a unha rede:

```
C:\Windows\system32>ping 192.168.56.105

Haciendo ping a 192.168.56.105 con 32 bytes de datos:
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.56.105: bytes=32 tiempo=1ms TTL=128

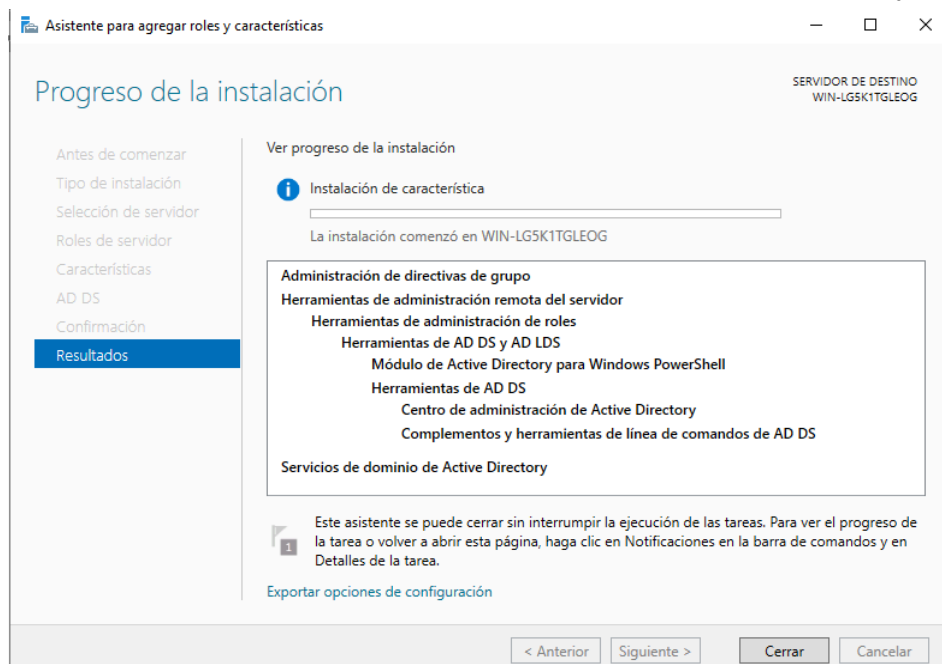
Estadísticas de ping para 192.168.56.105:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 0ms, Máximo = 1ms, Media = 0ms

C:\Windows\system32>_
```

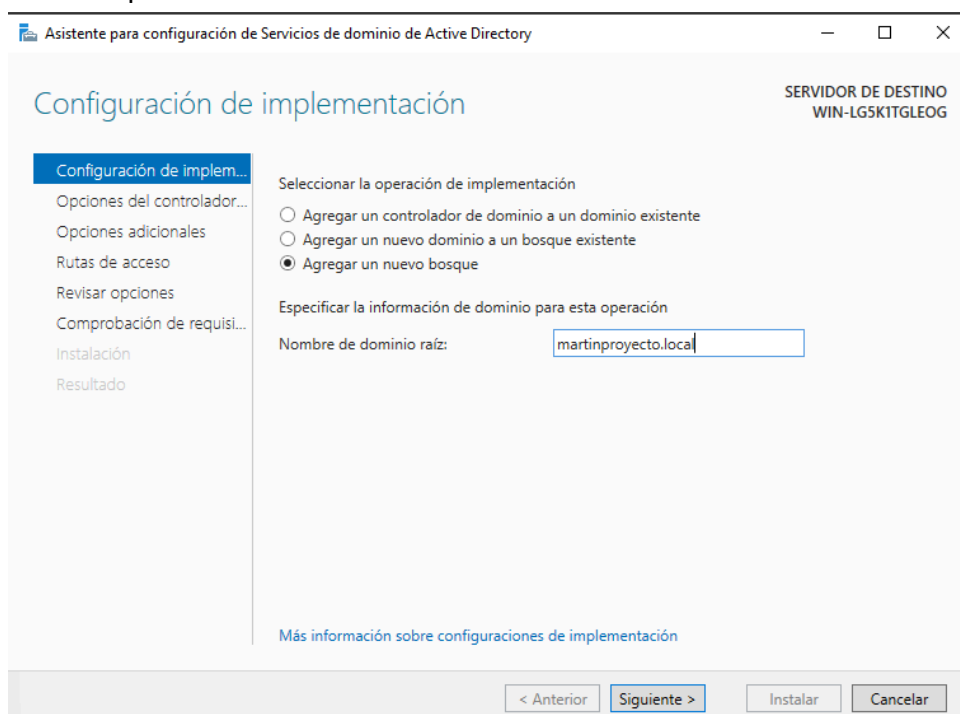
### 3. CONFIGURACIONES PREVIAS

Imos crear un dominio no Windows Server para meter o Cliente no mesmo.

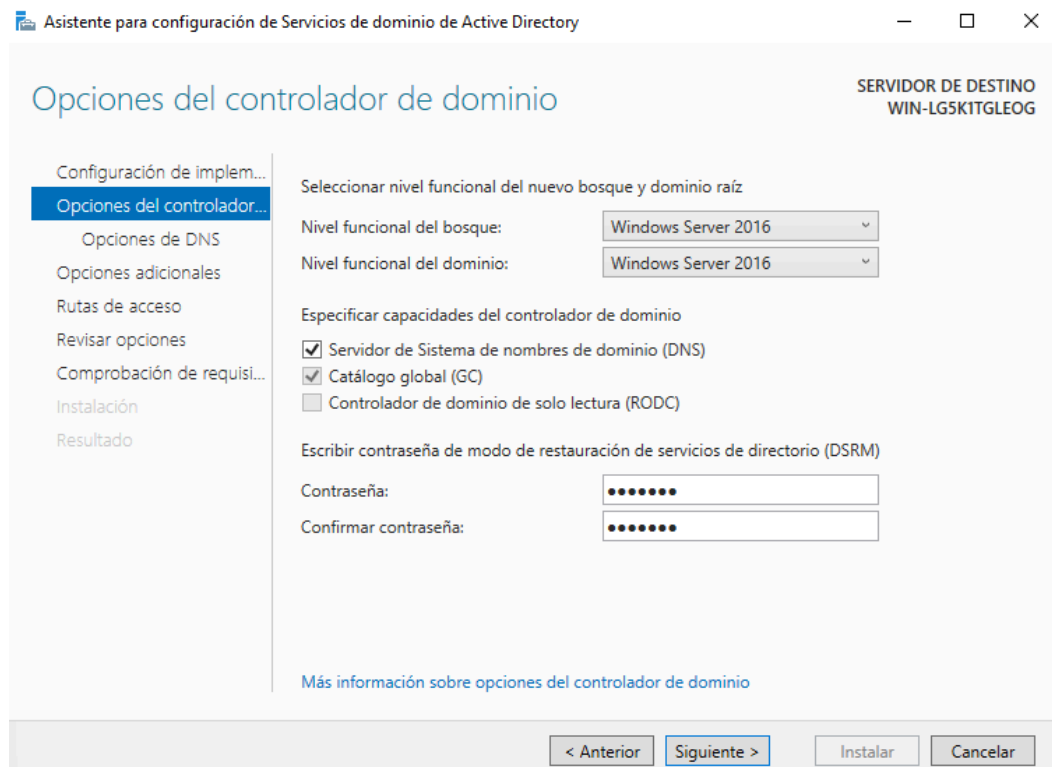
→ Para comenzar, temos que instalar os servicios de Active Directory



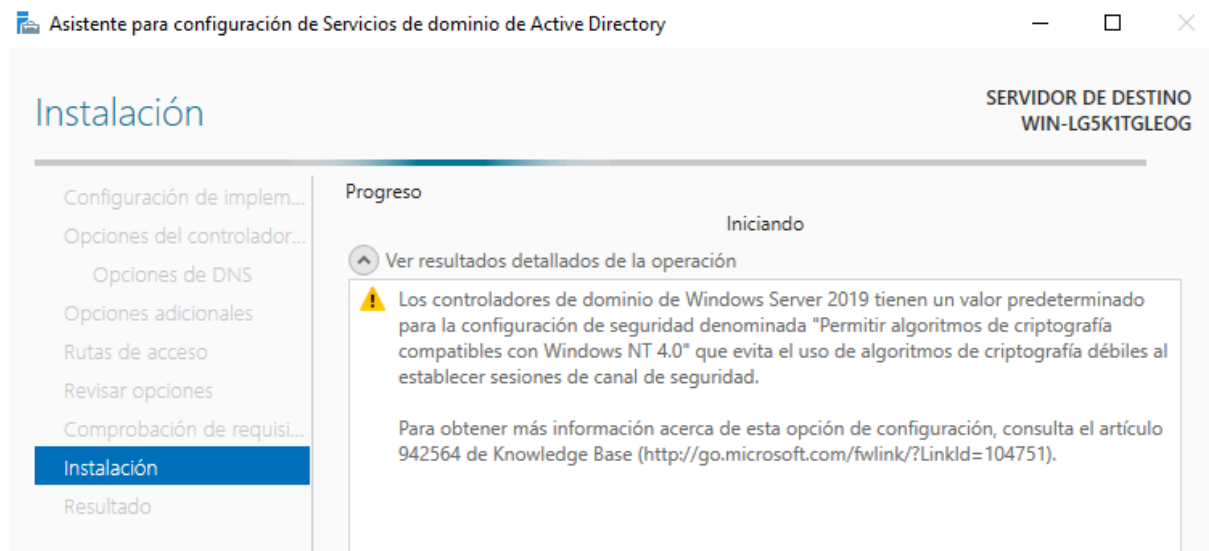
→ Ahora promovemos o Servidor a Servidor de Dominio e creamos o dominio



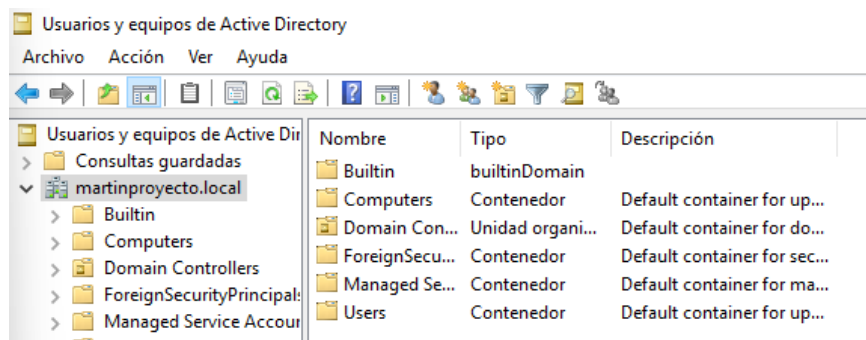
→ Seleccionamos o nivel funcional e marcamos a opción de DNS



→ Damosle a siguiente, e instalamos.



→ Imos comprobar que se creou correctamente:



→ Ahora temos que meter o Cliente Windows no dominio que acabamos de crear:

Cambios en el dominio o el nombre del equipo X

Puede cambiar el nombre y la pertenencia de este equipo. Los cambios podrían afectar al acceso a los recursos de red.

Nombre de equipo:  
pruebamartin

Nombre completo de equipo:  
pruebamartin

Más...

Miembro del


☒ Dominio:  
martinproyecto.local

☐ Grupo de trabajo:  
WORKGROUP

Aceptar Cancelar


→ Comprobamos:


Cambios en el dominio o el nombre del equipo X


 Se unió correctamente al dominio martinproyecto.local.

Aceptar


→ Ahora accedemos co Usuario ó cal lle vou aplicar as regras de AppLocker

 Cambiar la configuración de la cuenta


 Bloquear

 Cerrar sesión

---


 Cambiar de usuario

---

 pruebamartin

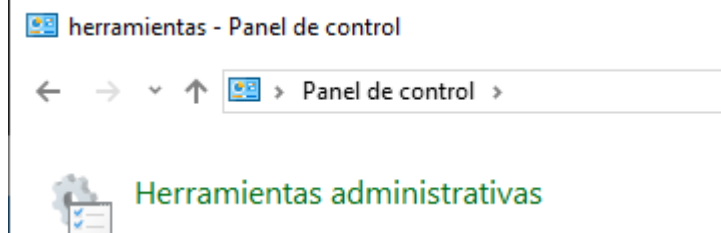
→ Comprobamos tamen no controlador de dominio que sí aparece:

Usuarios y equipos de Active Dir	
> Consultas guardadas	
▼ martinproyecto.local	
Built-in	
Computers	
Domain Controllers	
ForeignSecurityPrincipal	

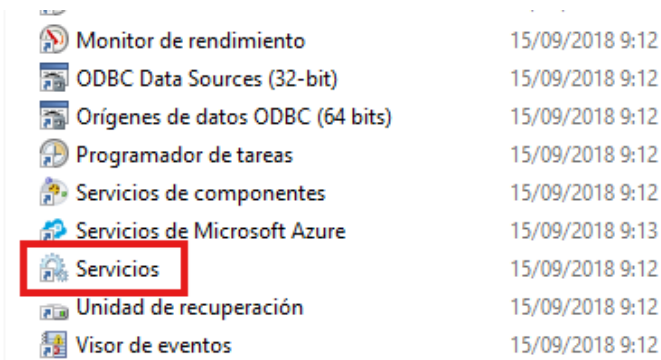
Nombre	Tipo
 PRUEBAMARTIN	Equipo

Para comenzar, AppLocker require que estea habilitado o servicio "Identidade de aplicacións" (Application Identity).

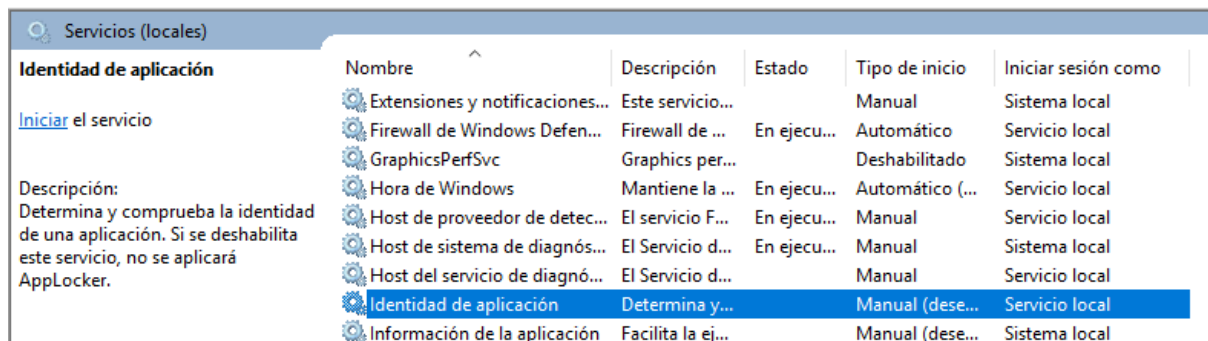
→ Imos habilitar o servicio "Application Identity". Temos que acceder o panel de control e acceder a ferramentas administrativas:



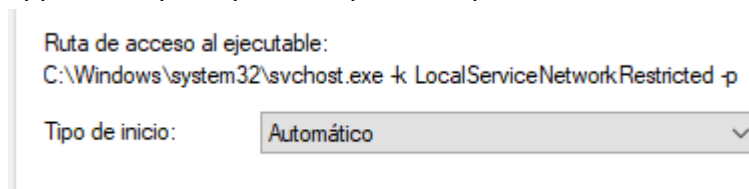
→ Unha vez dentro, accedemos a servicios:



→ Entramos a servicios, e buscamos identidade de aplicación:



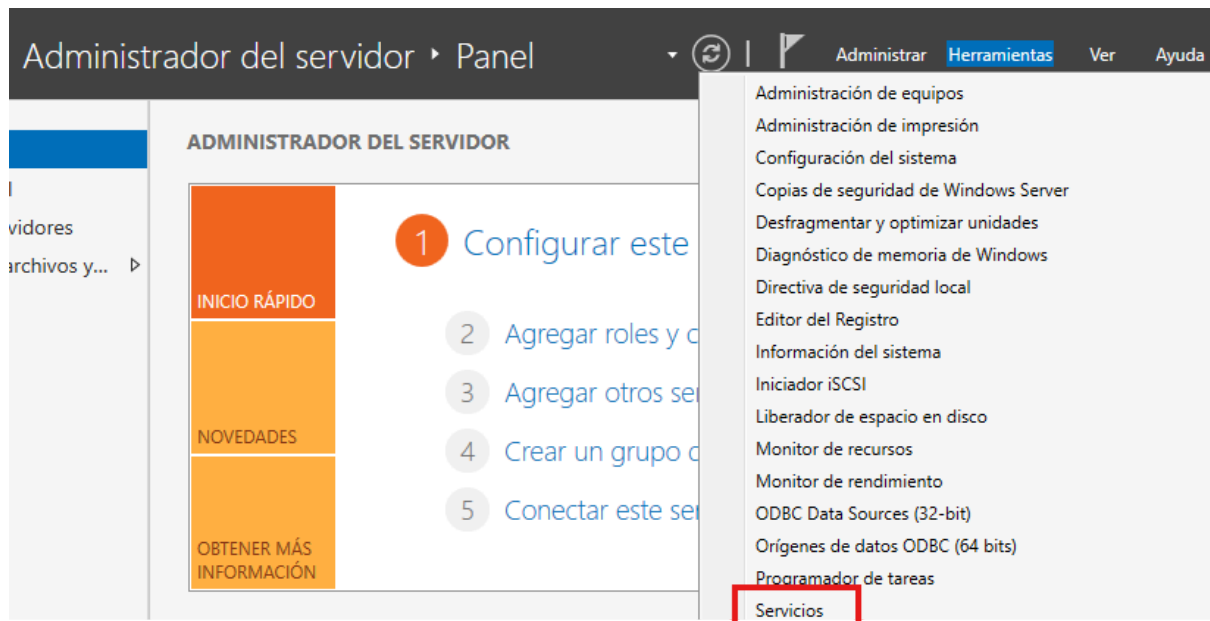
→ Como podemos ver, xa nos dí na descripción que se está deshabilitado non se aplicará a AppLocker, polo que imos poñerlle que se inicie automáticamente:



→ E agora imos inicialo:



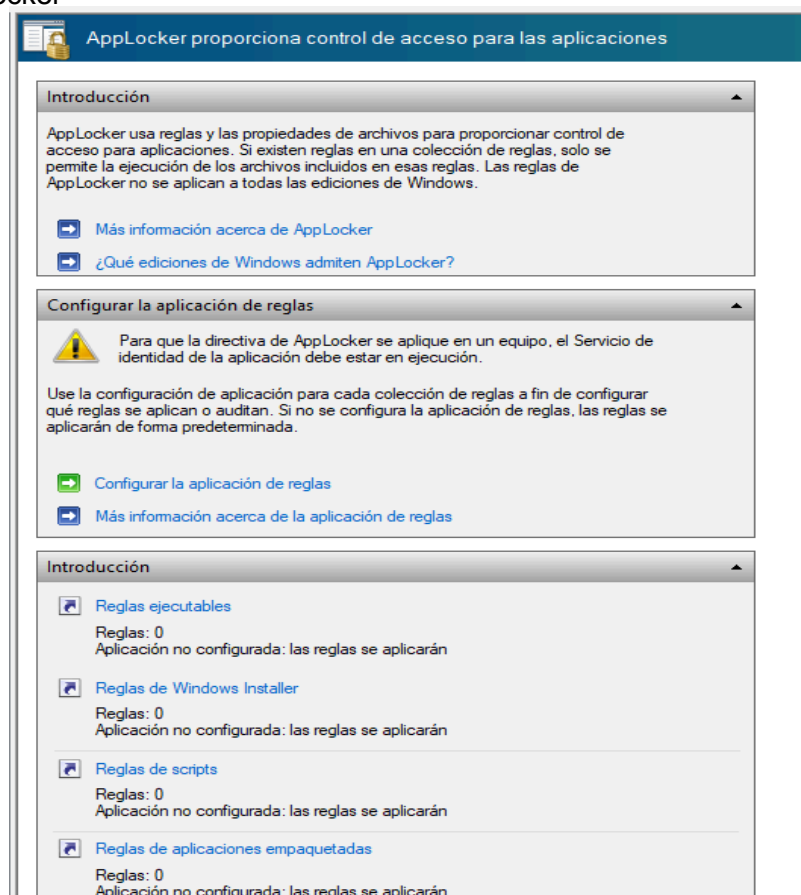
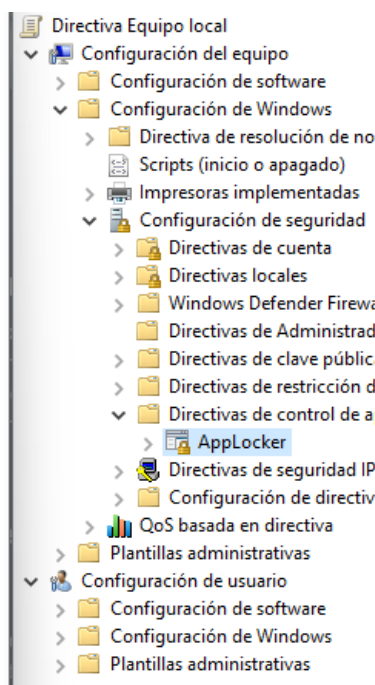
Tamén podríamos acceder directamente desde el Administrador del Servidor, en herramientas:



### AppLocker configúrase mediante políticas de grupo

→ Imos acceder a las políticas de grupo (GPOs) para llegar a AppLocker

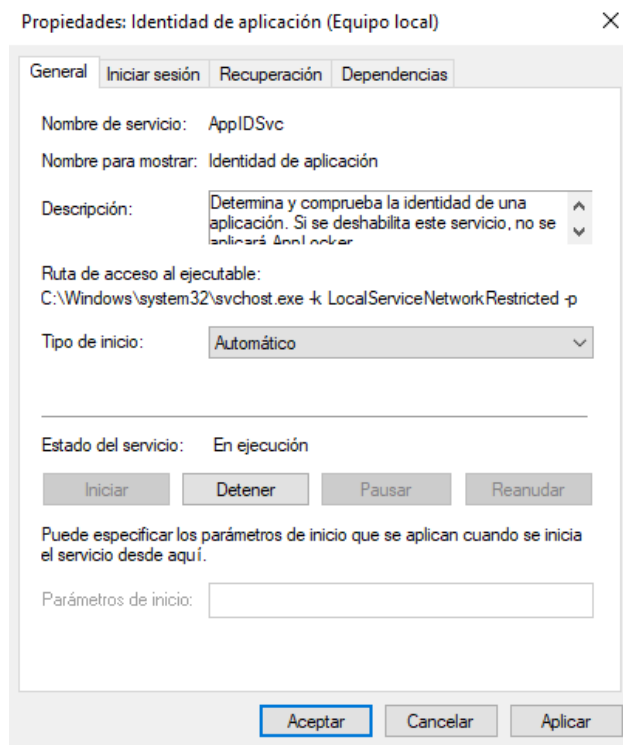
Configuración del equipo → Configuración de Windows → Configuración de seguridad → Directivas de control de aplicaciones → AppLocker





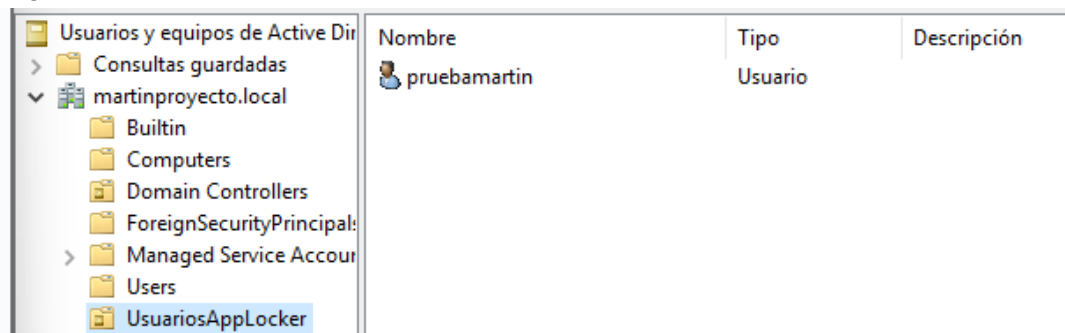
No cliente de Windows 10, tamén teremos que activar o servizo que falamos anteriormente, Identidade de Aplicación para poder aplicar AppLocker.

→ Procedemos:



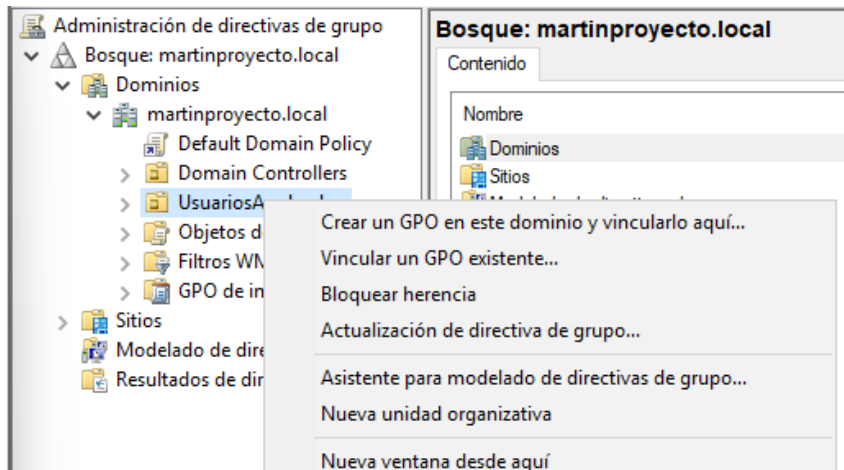
Iniciamos o servizo e poñemos en tipo de inicio Automático para que se inicie solo en cada reinicio.

→ Para que as GPOs se apliquen a un usuario do dominio, temos que crear unha Unidade Organizativa(OU) e introducir o usuario nesa unidade.



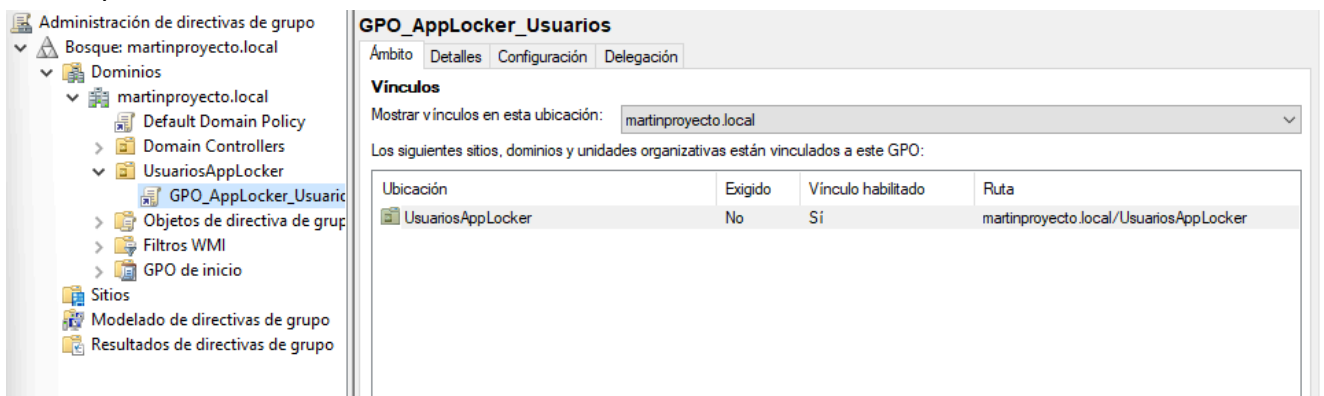
Tamén poderíamos introducir na Unidade Organizativa o equipo cliente, así aplicaríase a todos os usuarios que inicien sesión no equipo.

→ Ahora temos que crear unha GPO nesta Unidad Organizativa:



Accedemos a Administración de directivas de grupo de creamos unha GPO

→ Comprobamos:



Ahora comenzamos a creación das regras.

Para actualizar as regras no cliente, hai que introducir o comando gpupdate /force e comenzarán a actualizarse as directivas

```
C:\Windows\system32>gpupdate /force
Actualizando directiva...

La actualización de la directiva de equipo se completó correctamente.
Se completó correctamente la Actualización de directiva de usuario.
```

#### 4. CREACIÓN DAS REGLAS PARA HARDENING AVANZADO CON APPLOCKER

Comenzamos ca creación das regras:

- 1-Permitir só executables firmados por Microsoft en C:\Windows
- 2-Bloquear calquera .exe fóra de Program Files e Windows
- 3-Bloquear scripts (.ps1, .vbs, .js) fóra de rutas seguras
- 4-Bloquear ficheiros MSI e instaladores en rutas non autorizadas
- 5-Permitir solo scripts concedidos por Microsoft
- 6-Bloquear execución de aplicacións dende o cartafol do usuario
- 7-Bloquear cmd.exe, powershell.exe e wscript.exe para usuarios estándar

## 8-Prohibir execución de aplicacións a un usuario

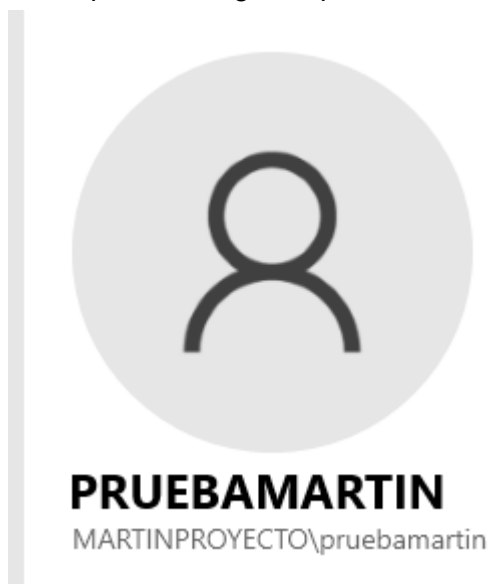
A través da implementación destas 8 regras de AppLocker, conseguimos establecer un control moi estrito e efectivo sobre a execución de aplicacións no sistema operativo Windows. Estas políticas permiten limitar drasticamente o riesgo de execución de software non autorizado, scripts maliciosos e instalacións non controladas, ao tempo que se garante a operatividade do sistema para os usuarios lexítimos.

AppLocker preséntase como unha ferramenta clave no bastionado de sistemas Windows, xa que proporciona un marco configurable e detallado para definir quen pode executar que tipo de aplicacións e dende onde. A súa aplicación correcta non solo minimiza a superficie de ataque, senón que tamén reforza as políticas de seguridade corporativas ao bloquear vectores comúns de malware e técnicas de persistencia.

Creáronse as regras predeterminadas de AppLocker como punto de partida:

Acción	Usuario	Nombre	Condición
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Archivos de programa	Ruta de acceso
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Windows	Ruta de acceso
✓ Permitir	BUILTIN\Ad...	(Regla predeterminada) Todos los archivos	Ruta de acceso

Para aplicar as regras e probalas vou utilizar o usuario do Dominio Pruebamartin



## 1. Permitir só executables asinados por Microsoft en C:\Windows

Con esta regra permitimos a execución de executables asinados por Microsoft unicamente na ruta do sistema, garantindo integridade dos binarios críticos.

→ Accedemos a regras executables e creamos unha nova regra:

Crear Reglas ejecutables

**Permisos**

Antes de comenzar

Permisos

Condiciones

Editor

Excepciones

Nombre

Seleccione la acción que desee usar y el usuario o grupo al que debe aplicarse esta regla. Una acción de permiso permite que los archivos afectados se ejecuten, mientras que una acción de denegación lo impide.

Acción:

☒ Permitir

☐ Denegar

Usuario o grupo:

MARTINPROYECTO\pruebamartin

Seleccionar...

→ En condición escollemos editor:

Antes de comenzar

Permisos

**Condiciones**

Editor

Excepciones

Nombre

Seleccione el tipo de condición principal que desee crear.

☒ Editor

Seleccione esta opción si la aplicación para la que desea crear la regla está firmada por el editor de software.

☐ Ruta de acceso

Cree una regla para una ruta de acceso de carpeta o archivo específica. Si selecciona una carpeta, la regla afectará a todos los archivos que contenga.

☐ Hash de archivo

Seleccione esta opción si desea crear una regla para una aplicación no firmada.

→ Ahora escollemos un executable de Microsoft

Archivo de referencia:

C:\Windows\System32\notepad.exe

Examinar...

Cualquier editor

Editor: O=MICROSOFT CORPORATION, L=REDMOND, S=\\

Nombre del producto: MICROSOFT® WINDOWS® OPERATING SYSTEM

Nombre de archivo: \*

Versión del archivo: \*

Y superior

Deixei en branco Nome do arquivo e Versión para permitir calquera ficheiro do sistema asinado por Microsoft.

Non vou crear ningunha excepción xa que solo estou restrinxindo a execución a C:\Windows

→ Comprobamos que se creou correctamente:

Acción	Usuario	Nombre	Condición
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Ar...	Ruta de a...
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Wi...	Ruta de a...
✓ Permitir	MARTINPROYECTO\pruebama...	MICROSOFT® WINDOWS® OPERATING SYSTEM, desde O...	Editor
✓ Permitir	BUILTIN\Administradores	(Regla predeterminada) Todos los archivos	Ruta de a...

Así evitamos que executables maliciosos enmascarados se ejecuten desde esa ruta clave.

## 2. Bloquear calquera .exe fóra de Program Files e Windows

Denegamos a execución de calquera ficheiro coa extensión .exe que se atope dentro de calquera subcarpeta do directorio C:\Users\\*\AppData\Local\\* ou directamente dentro da ruta C:\Temp\\*

→ Creamos a regra:

Seleccione la acción que desee usar y el usuario o grupo al que debe aplicarse esta regla. Una acción de permiso permite que los archivos afectados se ejecuten, mientras que una acción de denegación lo impide.

Acción:

☐ Permitir

☒ Denegar

Usuario o grupo:

MARTINPROYECTO\pruebamartin

Seleccionar...

→ Condición:

Seleccione el tipo de condición principal que desee crear.

☐ Editor

Seleccione esta opción si la aplicación para la que desea crear la regla está firmada por el editor de software.

☒ Ruta de acceso

Cree una regla para una ruta de acceso de carpeta o archivo específica. Si selecciona una carpeta, la regla afectará a todos los archivos que contenga.

☐ Hash de archivo

Seleccione esta opción si desea crear una regla para una aplicación no firmada.

→ En ruta de acceso, poñemos C:\

Seleccione la ruta de acceso de archivo o de carpeta a la que debe afectar esta regla. Si especifica una ruta de acceso de carpeta, esta regla afectará a todos los archivos ubicados en esa ruta de acceso.

Ruta de acceso:

Examinar archivos... Examinar carpetas...

→ Ahora en excepciones poñemos as Rutas de Program Files e Windows:

Condición principal:  
%OSDRIVE%\\*

Agregar excepción:  
Ruta de acceso

Excepciones:

Excepción	Tipo
%PROGRAMFILES%\*	Ruta de acceso
%WINDIR%\	Ruta de acceso

Agregar... Editar Quitar

→ Comprobamos:

Acción	Usuario	Nombre	Condición	Excepcio...
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Ar...	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Wi...	Ruta de a...	
✓ Permitir	MARTINPROYECTO\pruebama...	MICROSOFT® WINDOWS® OPERATING SYSTEM, desde O...	Editor	
✓ Permitir	BUILTIN\Administradores	(Regla predeterminada) Todos los archivos	Ruta de a...	
✗ Denegar	MARTINPROYECTO\pruebama...	Bloqueo EXE fora de rutas estándar	Ruta de a...	Sí

Así bloqueamos a ejecución de posibles malware que se descarguen e garden en rutas comúns do usuario, como o escritorio ou descargas

### 3. Bloquear scripts (.ps1, .vbs, .js) fóra de rutas seguras

Esta regla evita que os scripts maliciosos se executen desde carpetas do usuario

→ Para crear esta regla, temos que facelo en Reglas de Script, antes de nada imos crear as Reglas Predeterminadas de este apartado

→ Imos crear a nova regra: Acción, denegar

Seleccione la acción que desee usar y el usuario o grupo al que debe aplicarse esta regla. Una acción de permiso permite que los archivos afectados se ejecuten, mientras que una acción de denegación lo impide.

Acción:

☐ Permitir

☒ Denegar

Usuario o grupo:

MARTINPROYECTO\pruebamartin

Seleccionar...

→ Condición:

Seleccione el tipo de condición principal que desee crear.

☐ Editor

Seleccione esta opción si la aplicación para la que desea crear la regla está firmada por el editor de software.

☒ Ruta de acceso

Cree una regla para una ruta de acceso de carpeta o archivo específica. Si selecciona una carpeta, la regla afectará a todos los archivos que contenga.

☐ Hash de archivo

Seleccione esta opción si desea crear una regla para una aplicación no firmada.

→ Ruta:

Seleccione la ruta de acceso de archivo o de carpeta a la que debe afectar esta regla. Si especifica una ruta de acceso de carpeta, esta regla afectará a todos los archivos ubicados en esa ruta de acceso.

Ruta de acceso:

%OSDRIVE%\Users\\*

Examinar archivos...

Examinar carpetas...

→ Neste caso, non crearemos ningunha excepción

→ Comprobamos:



Acción	Usuario	Nombre	Condición	Excepcio...
✓ Permitir	MARTINPROYECTO\pruebamartin	Permitir scripts solo firmados por Micro...	Editor	
✓ Permitir	Todos	(Regla predeterminada) Todos los script...	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los script...	Ruta de a...	
✓ Permitir	BUILTIN\Administradores	(Regla predeterminada) Todos los scripts	Ruta de a...	
✗ Denegar	MARTINPROYECTO\pruebamartin	Bloqueo de scripts en perfiles de usuario	Ruta de a...	

Así defendemos ataques por script (por ejemplo PowerShell malicioso) ejecutados desde descargas, escritorio, etc.

#### 4. Bloquear ficheros MSI e instaladores en rutas no autorizadas

Con esta regla de AppLocker impedimos a ejecución de ficheros .msi (instaladores de Windows) que se atopen dentro de cualquiera carpeta de usuario en %OSDRIVE%\Users\\*

→ Creamos la regla, neste caso vai ser en Windows Installer polo que imos crear as reglas predeterminadas

→ En acción, denegamos:

Seleccione la acción que desee usar y el usuario o grupo al que debe aplicarse esta regla. Una acción de permiso permite que los archivos afectados se ejecuten, mientras que una acción de denegación lo impide.

Acción:

☐ Permitir

☒ Denegar

Usuario o grupo:

MARTINPROYECTO\pruebamartin Seleccionar...

→ En condición:

Seleccione el tipo de condición principal que desee crear.

☐ Editor

Seleccione esta opción si la aplicación para la que desea crear la regla está firmada por el editor de software.

☒ Ruta de acceso

Cree una regla para una ruta de acceso de carpeta o archivo específica. Si selecciona una carpeta, la regla afectará a todos los archivos que contenga.

☐ Hash de archivo

Seleccione esta opción si desea crear una regla para una aplicación no firmada.

→ Ruta: Carpeta de usuarios

Ruta de acceso:

→ Non creamos ningunha excepción

→ Comprobamos:

Acción	Usuario	Nombre	Condición	Excepcio...
Permitir	Todos	(Regla predeterminada) Todos los archivos d...	Editor	
Permitir	Todos	(Regla predeterminada) Todos los archivos d...	Ruta de acceso	
Permitir	BUILTIN\Ad...	(Regla predeterminada) Todos los archivos d...	Ruta de acceso	
Denegar	MARTINPRO...	Bloqueo de instaladores a usuarios	Ruta de acceso	

Así reducimos o riesgo de instalación de software non controlado (e posiblemente peligroso)

## 5. Permitir solo scripts concedidos por Microsoft

Esta regla permite a execución solo de scripts de confianza concedidos/permitidos digitalmente por Microsoft.

→ Volvemos as Reglas de Scripts e creamos unha nova:

**Permisos**

Antes de comenzar

**Permisos**

Condiciones

Editor

Excepciones

Nombre

Seleccione la acción que desee usar y el usuario o grupo al que debe aplicarse esta regla. Una acción de permiso permite que los archivos afectados se ejecuten, mientras que una acción de denegación lo impide.

Acción:

☒ Permitir

☐ Denegar

Usuario o grupo:

MARTINPROYECTO\pruebamartin

Seleccionar...

→ Condición:

Seleccione el tipo de condición principal que desee crear.

☒ Editor

Seleccione esta opción si la aplicación para la que desea crear la regla está firmada por el editor de software.

☐ Ruta de acceso

Cree una regla para una ruta de acceso de carpeta o archivo específica. Si selecciona una carpeta, la regla afectará a todos los archivos que contenga.

☐ Hash de archivo

Seleccione esta opción si desea crear una regla para una aplicación no firmada.

→ En Editor, imos seleccionar un executable de microsoft, como por exemplo notepad.exe .  
Despois seleccionamos a nivel de editor (Microsoft Corporation) e o resto deixamolo en \* para calquera ficheiro.

→ Non añadimos ningunha excepción

→ Comprobamos:

Acción	Usuario	Nombre	Condición	Excepcio...
✓ Permitir	BUILTIN\Administradores	(Regla predeterminada) Todos los scripts	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los script...	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los script...	Ruta de a...	
✗ Denegar	MARTINPROYECTO\pruebamartin	Bloqueo de scripts en perfiles de usuario	Ruta de a...	
✓ Permitir	MARTINPROYECTO\pruebamartin	Permitir scripts solo firmados por Micro...	Editor	

Así solo se permiten scripts oficiais e firmados por Microsoft, evitando PowerShells maliciosos

## 6. Bloquear ejecución de aplicacións dende o cartafol do usuario

Bloquear calquera .exe que se copie ou descargue no escritorio, descargas, documentos ou outras rutas do perfil de usuario, que é onde normalmente se gardan aplicacións portables ou maliciosas.

→ En reglas ejecutables, introducimos a accion:

→ En condicións, ruta de acceso

→ Ruta de acceso:

Ruta de acceso:

→ En excepcións poderíamos añadir algunha se fose necesaria para ese perfil, como por exemplo, aplicacións de Edge ou algo polo estilo

\Users\NomeUsuario\Microsoft\Edge\Application

No meu caso, non vou añadir ningunha

→ Comprobamos:

Acción	Usuario	Nombre	Condición	Excepcio...
✓ Permitir	BUILTIN\Administradores	(Regla predeterminada) Todos los archivos	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Ar...	Ruta de a...	
✓ Permitir	Todos	(Regla predeterminada) Todos los archivos de la carpeta Wi...	Ruta de a...	
✓ Permitir	MARTINPROYECTO\pruebama...	MICROSOFT® WINDOWS® OPERATING SYSTEM, desde O...	Editor	
✗ Denegar	MARTINPROYECTO\pruebama...	Bloquear ejecutables nas rutas do usuario	Ruta de a...	
✗ Denegar	MARTINPROYECTO\pruebama...	Bloqueo EXE fora de rutas estándar	Ruta de a...	Sí

Así evitamos que os usuarios executen software portable descargado ou copiado localmente nas súas carpetas personales, unha das vías máis habituais de entrada de malware e ferramentas maliciosas

## 7. Bloquear cmd.exe, powershell.exe e wscript.exe para usuarios estándar

Evitamos que usuarios sin privilexios administrativos usen ferramentas de sistema peligrosas.

→ Seguimos en regras ejectubales. Applocker non permite crear unha regra con tres rutas de acceso diferentes, polo tanto, teremos que crear tres regras, unha para cada ejecutable

→ Denegamos permisos e aplicamosllo ó usuario *pruebamartin*

→ Condicións, ruta de acceso

Ruta de acceso:





  

Ruta de acceso:

Ruta de acceso:

Ruta de acceso:

→ Así xa estarían bloqueados os 4 executables dos programas peligrosos. Comprobamos:

	Denegar	MARTINPROYECTO\pruebama...	Restringir cmd.exe a usuarios	Ruta de a...
	Denegar	MARTINPROYECTO\pruebama...	Restringir powershell.exe a usuarios	Ruta de a...
	Denegar	MARTINPROYECTO\pruebama...	Restringir powershell_ise.exe a usuarios	Ruta de a...
	Denegar	MARTINPROYECTO\pruebama...	Restringir wscript.exe a usuarios	Ruta de a...

Con esto previmos que usuarios usen PowerShell ou CMD para executar comandos maliciosos.

## 8. Prohibir ejecución de aplicaciones a un usuario

Esta regra prohibe o uso de aplicacións non deseadas para así reducir a posibilidade de peligro. No meu caso imos prohibir Internet Explorer.


→ Creamos a regra nova, denegamos permisos e aplicamoslla ó usuario

Seleccione la ruta de acceso de archivo o de carpeta a la que debe afectar esta regla. Si especifica una ruta de acceso de carpeta, esta regla afectará a todos los archivos ubicados en esa ruta de acceso.

Ruta de acceso:

Tamén vou facer o mesmo, por exemplo, para o Calendario ou a Calculadora

→ Creamos a regra e comprobamos que se creou:

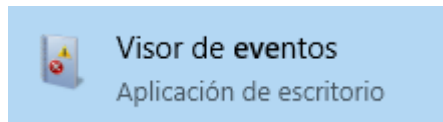
	Denegar	MARTINPROYECTO\pruebama...	Prohibir uso de iexplore.exe	Ruta de a...
---	---------	----------------------------	------------------------------	--------------

Así garantimos que no se utiliza la aplicación, manteniendo seguridad.

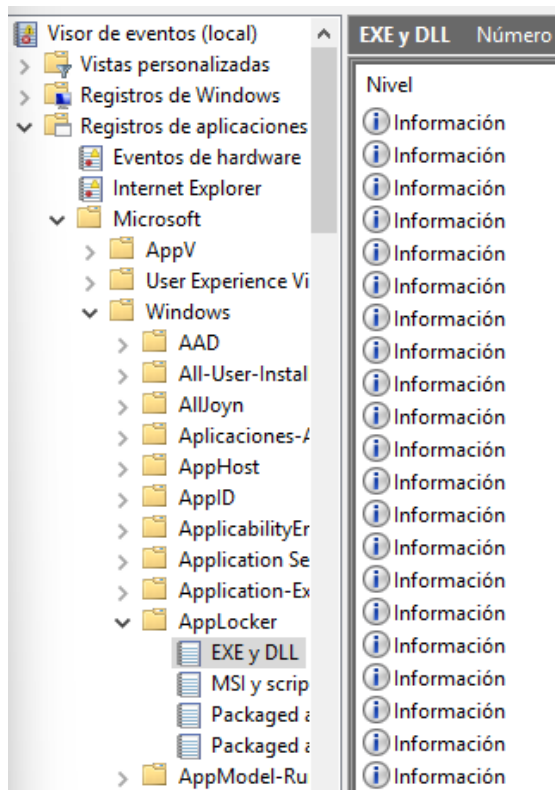
## 5. VISOR DE EVENTOS DE APPLOCKER

Permite auditar la ejecución de aplicaciones, comprobar que reglas se están aplicando e detectar intentos de ejecución bloqueados.

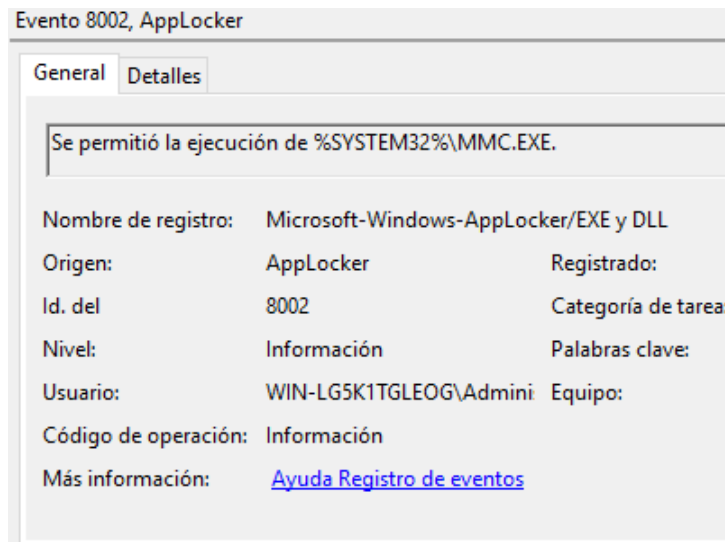
Para acceder tenemos que abrir el visor de eventos



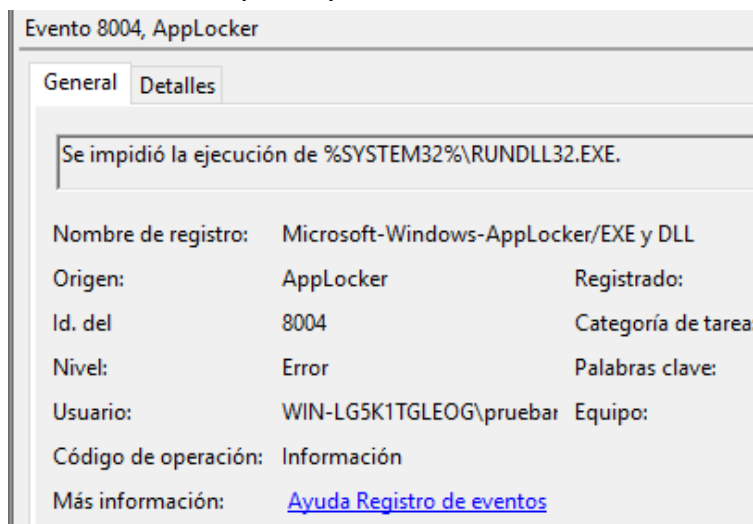
Una vez dentro, Registros de aplicaciones y servicios > Microsoft > Windows > AppLocker > EXE y DLL



Aquí podemos ver todos los eventos de información e incluso de intentos de acceso  
**Eventos de información ( 8002 ):**



### Eventos de error( 8004 ):



### Para que se utiliza:

- Verificar se as regras funcionan correctamente.
- Identificar intentos de ejecución bloqueados por parte dos usuarios.
- Detectar posibles tentativas de ejecución de malware ou software non autorizado.
- Axudar no axuste fino das políticas sen romper o sistema.

## 6. PROBA REGLAS DE APPLOCKER

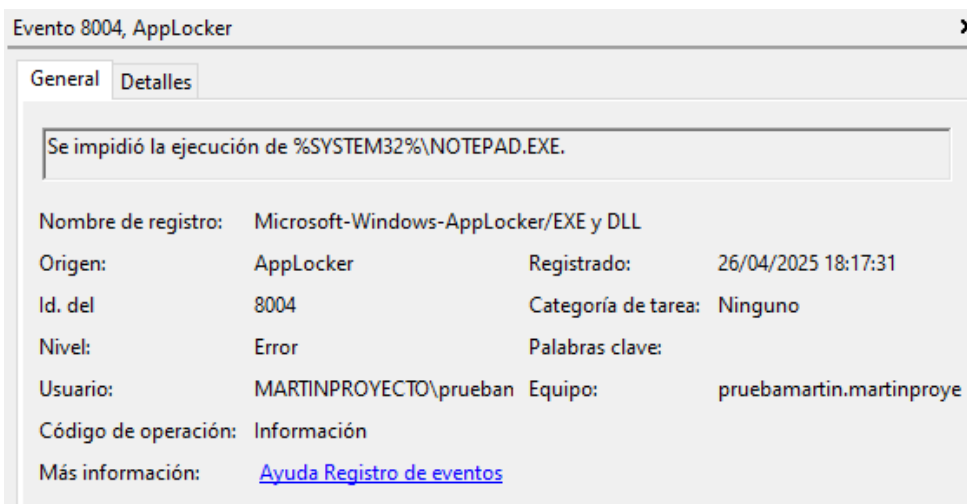
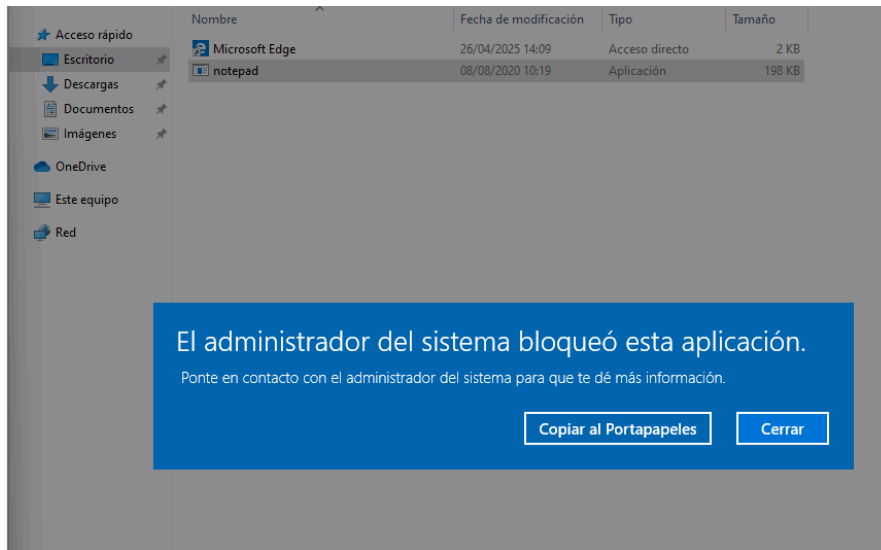
### Comenzamos ca proba das regras:

**1-Permitir só executables asinados por Microsoft en C:\Windows**  
[video solo firmados microsoft](#)



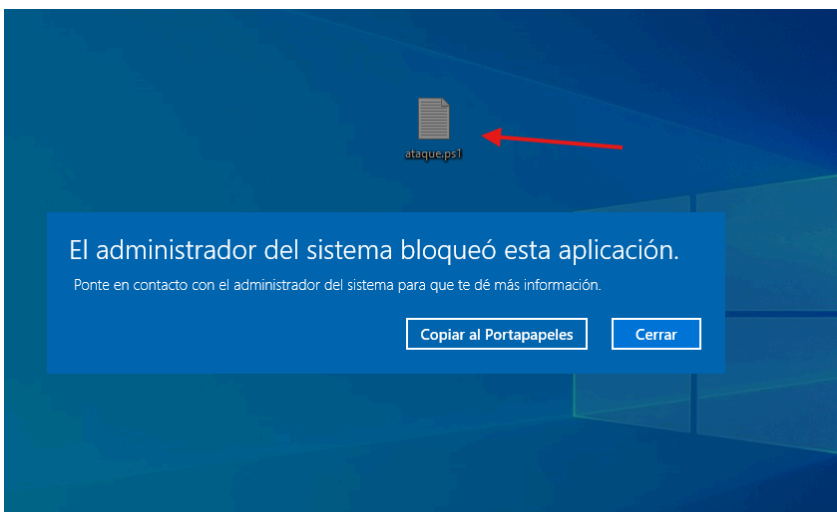
## 2-Bloquear calquera .exe fóra de Program Files e Windows

[video .exe fora de Program Files e Windows](#)



## 3-Bloquear scripts (.ps1, .vbs, .js) fóra de rutas seguras

[video bloqueo scripts](#)



#### 4-Bloquear ficheiros MSI e instaladores en rutas non autorizadas

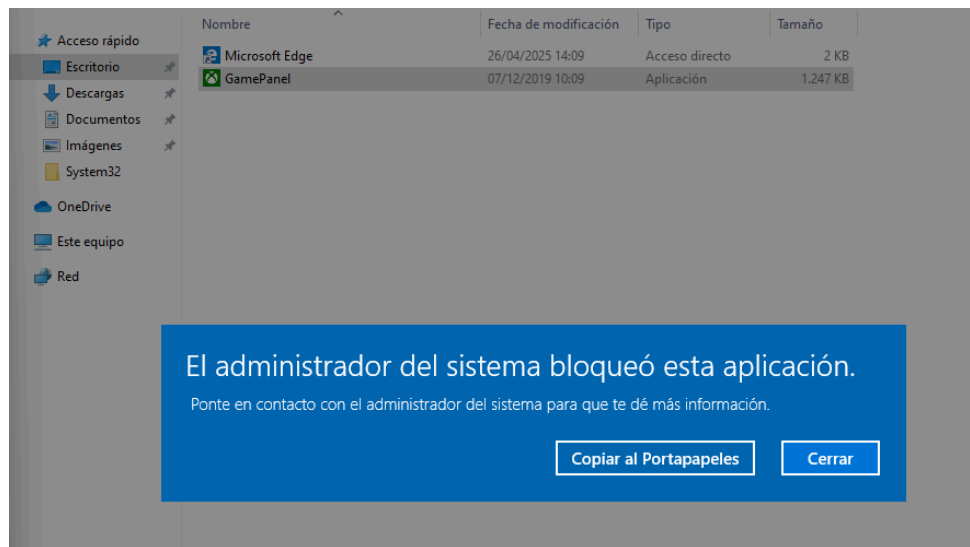
[video Bloqueo MSI](#)

#### 5-Permitir solo scripts concedidos por Microsoft

[video scripts concedidos Microsoft](#)

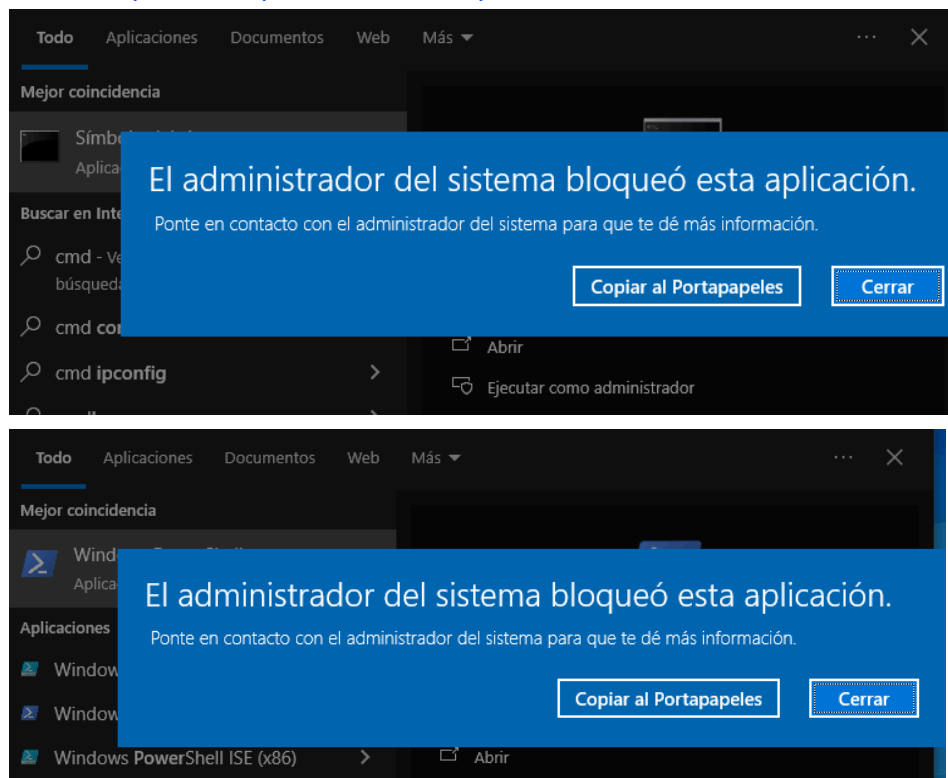
#### 6-Bloquear execución de aplicacións dende o cartafol do usuario

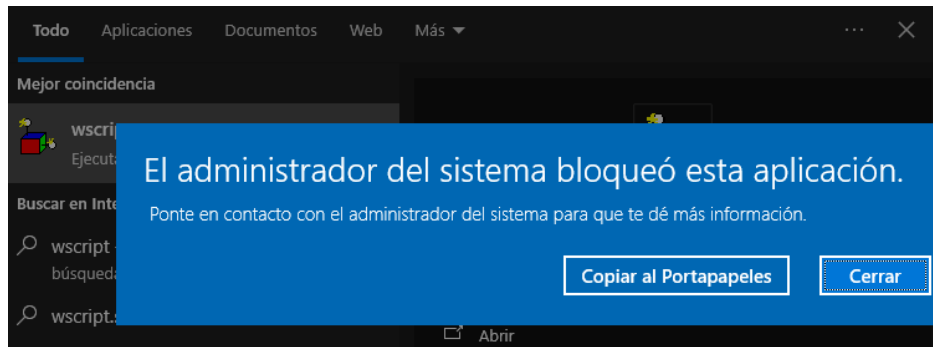
[video aplicacion desde a ruta do usuario](#)



#### 7-Bloquear cmd.exe, powershell.exe e wscript.exe para usuarios

[video bloqueo cmd-powershell-wscript](#)





## 8-Prohibir ejecución de aplicaciones a un usuario

[vídeo prohibir aplicaciones](#)

# 6. BIBLIOGRAFÍA / WEBGRAFÍA

(Principal) Aprendizaxe obtido en Bastionado de Redes e Sistemas e tamén do resto de módulos do Máster de Ciberseguridade

<https://www.elladodelmal.com/2020/01/servicios-ssh-tips-de-auditoria-y.html>

[https://docs.redhat.com/es/documentation/red\\_hat\\_enterprise\\_linux/8/html-single/using\\_selinux/index](https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html-single/using_selinux/index)

<https://www.tarlogic.com/es/blog/selinux-para-proteger-una-web/>

<https://puerto53.wordpress.com/2021/08/08/guia-basica-de-selinux/>

<https://books.spartan-cybersec.com/cpad/introduccion-a-la-evasion-de-defensas/introduccion-a-applocker>

<https://www.asnet.es/utilizacion-applocker-bloquear-aplicaciones-sobre-windows-10-servidor-de-rds/>

<https://github.com/DarbyShin/Windows-10-IoT-Enterprise/blob/master/Step%20by%20Step%20Guide%20-%20Windows%2010%20AppLocker%20Guide.pdf>

<https://learn.microsoft.com/es-es/windows/security/application-security/application-control/app-control-for-business/applocker/working-with-applocker-rules>